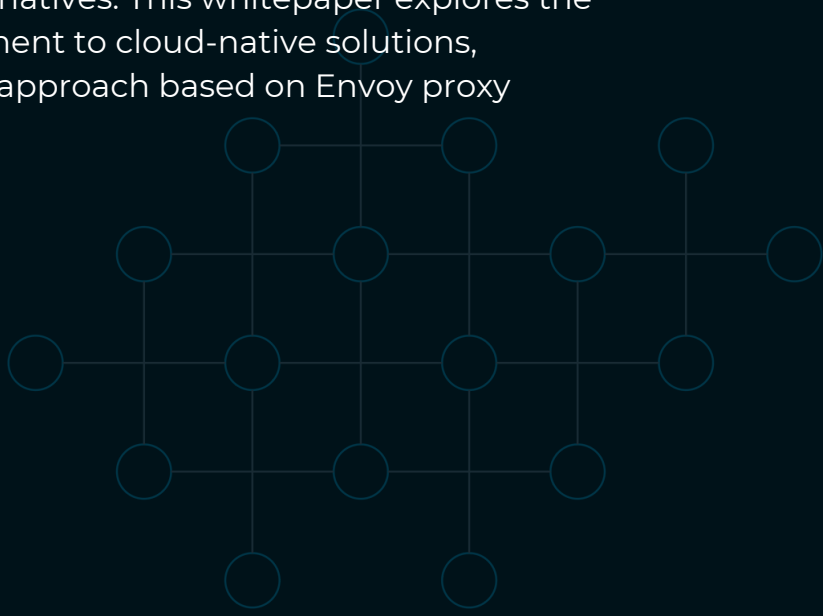
Evolution in API Management:

Cloud-Native Solutions and Future-Proofing with Solo.io Gloo Gateway

WHITE PAPER

Abstract

As the landscape of software development evolves, organizations face the challenge of managing a growing number of microservices while adopting platform engineering best practices, workflows, and modern API management tools. Google's Apigee has been a dominant force in the traditional API management space, but the emergence of cloud-native solutions like Solo.io's Gloo Gateway presents compelling alternatives. This whitepaper explores the transition from traditional API management to cloud-native solutions, highlighting the advantages of Solo.io's approach based on Envoy proxy technology.



Introduction

The proliferation of microservices has transformed the way modern development teams operate. Organizations are increasingly adopting platform engineering best practices and relying on API management tools to streamline their development workflows. Google's Apigee has been a longstanding player in the API management domain, offering a comprehensive suite of tools for designing, testing, and deploying APIs. However, these “all-in-one” stacks don’t fit modern workflows and expectations around automation. The shift towards cloud-native architectures and the need for greater flexibility and scalability have prompted organizations to explore alternative solutions.

In this white paper, we delve into the challenges faced by organizations dealing with the explosion of microservices and the complexities of managing API lifecycles. We examine the evolution from traditional API management using proxies such as Envoy, and proprietary offerings such as ApigeeX and provide guidance as to what platform teams should be considering as they look to move to cloud-native solutions. We will also showcase the benefits of Solo.io's Gloo Gateway, built on a CNCF open source project, Envoy proxy, as a forward-looking alternative for modern development environments.

The Transition to Cloud-Native API Management

Managing APIs in the context of the escalating microservices landscape presents a range of formidable challenges. The complexity mounts as numerous APIs must navigate diverse technologies and data formats across distributed microservices architectures, straining integration efforts. Maintaining robust API governance, lifecycle management, scalability, and performance becomes paramount amidst the rapid proliferation of APIs and escalating traffic demands. Security concerns loom large, necessitating stringent measures against unauthorized access, data breaches, and compliance risks. When evaluating modernizing API management capabilities, platform teams must take the following into consideration as they look to future proof their investments.

- **Service decomposition:** Breaking down monolithic applications into hundreds and thousands of independent services.
- **Increased granularity:** Allowing developers to work on individual services independently, facilitating faster development cycles, easier maintenance, and scalability.
- **API proliferation:** With each microservice exposing its own API, the number of APIs within an organization grows exponentially as more services are added to the ecosystem. Future proofing for increased traffic and scalability can help organizations protect themselves from complex challenges and skyrocketing costs.
- **Diverse technologies:** Microservices are often developed using diverse technologies, frameworks, and programming languages based on the specific requirements of each service. This diversity adds another layer of complexity as organizations must manage APIs that are built on different technologies and may have varying standards and protocols.
- **Scalability and agility:** While microservices offer scalability and agility benefits, the sheer number of APIs can pose challenges in scaling infrastructure, handling increased traffic, and maintaining agility in development and deployment processes.

Evolution of API Management Tools

Envoy is fast becoming a popular tool for modern software architectures, particularly for handling microservices communication and API traffic. While they serve similar purposes, there are key differences in their design, features, and use cases. We have taken a deeper dive into the features and challenges related to each of these proxies as well as proprietary tools such as ApigeeX.

Envoy

Origin: Envoy was developed by Lyft in 2016 as a modern, cloud-native proxy server designed specifically for microservices architectures and containerized environments.

Use cases: Envoy is purpose-built for handling service-to-service communication in microservices architectures. It is also used as an edge proxy for managing incoming and outgoing traffic to/from microservices.

Features:

- **Cloud-native design:** Envoy is built from the ground up with cloud-native principles, making it highly adaptable to dynamic, distributed environments like Kubernetes, GKE, AKS, and EKS.
- **Service mesh:** Envoy is a core component in service mesh architectures (e.g., Istio, Linkerd) for managing east-west traffic between microservices, providing features like load balancing, circuit breaking, retries, and observability.
- **Dynamic configuration:** Envoy supports dynamic configuration updates, allowing it to adapt to changing network conditions, route configurations, and service discovery without downtime.
- **Extensibility:** Envoy offers a rich set of filters and plugins for extending its functionality, enabling developers to customize behavior for specific use cases (e.g., rate limiting, authentication, telemetry).
- **Configuration:** Envoy's configuration is typically done using YAML or JSON files, with support for dynamic configuration APIs like xDS (e.g., ADS, RDS, CDS) for real-time updates and control plane integration.

Challenges:

- **Complex configuration:** Envoy's configuration can be complex due the various options available. Managing and maintaining a large number of configuration files for routing, load balancing, retries, timeouts, etc., can become challenging and require expertise.
- **Learning curve:** Envoy has a steep learning curve for administrators and developers who are new to the technology.
- **Resource consumption:** Envoy is known for its performance and efficiency, however proper resource allocation and monitoring are essential to ensure optimal performance and avoid resource contention.
- **Dynamic updates:** While Envoy supports dynamic configuration updates through APIs like xDS (e.g., SDS, CDS, RDS), managing and orchestrating these updates in real-time can be complex.
- **Service discovery and health checking:** Envoy relies on service discovery mechanisms (e.g., DNS, Consul, etc) to discover backend services and perform health checks. Ensuring accurate and up-to-date service discovery and health statuses is crucial.

Solo.io's Gloo Gateway is designed to address the challenges associated with API management in cloud-native environments, including those related to proxy configuration, service discovery, observability, plugin management, and integration with API gateway features. Here's how Gloo Gateway helps address these challenges:

- **Gloo's configuration API**: Gloo Gateway provides a dynamic configuration API that allows for real-time updates to routing rules, load balancing configurations, security policies, and more.
- **Learning curve**: Gloo Gateway offers a simplified configuration model using a declarative configuration format (YAML or JSON) and user-friendly APIs. This reduces the learning curve for administrators and developers, making it easier to understand and manage Gloo's features and configurations.
- **Resource consumption**: Gloo Gateway is designed for efficient resource utilization, optimizing CPU and memory usage while maintaining high performance and scalability. It is well-suited for deployments with a large number of proxies or high traffic volumes in cloud-native environments.

- **Dynamic updates:** Gloo Gateway supports real-time configuration updates through its configuration API and integration with service discovery mechanisms like Kubernetes, Consul, and AWS App Mesh. Changes to routing rules, service endpoints, and policies can be applied dynamically without service disruption.
- **Service discovery integration with service mesh:** Gloo Gateway integrates seamlessly with service mesh solutions like Istio and Linkerd, leveraging their service discovery and health checking capabilities. This ensures accurate and up-to-date service discovery and enables advanced traffic management features like circuit breaking, retries, and fault injection.
- **Built-in observability:** Gloo Gateway provides built-in metrics, logging, and tracing capabilities for monitoring API traffic, performance metrics, error rates, and latency. It integrates with popular observability platforms like Prometheus, Grafana, Jaeger, and Zipkin for comprehensive visibility into API traffic and system health.
- **Extensive plugin support:** Gloo Gateway offers an extensive plugin ecosystem for extending and customizing API management functionalities. Plugins cover a wide range of features including authentication, rate limiting, transformations, security policies, caching, and more, allowing organizations to tailor Gloo to their specific requirements.
- **API gateway capabilities:** Gloo Gateway provides robust API gateway capabilities including API lifecycle management, developer portals such as the increasingly popular project Backstage, analytics, and monetization features. It seamlessly integrates with Gloo's plugin ecosystem and external tools for comprehensive API management in cloud-native environments.

Solo.io's Gloo Gateway addresses API management challenges in cloud-native environments by offering dynamic configuration, simplified management, efficient resource utilization, seamless integration with service mesh, built-in observability, an extensive plugin ecosystem, and comprehensive API gateway features. These capabilities make Gloo Gateway a versatile and effective solution for organizations adopting cloud-native practices and seeking robust API management capabilities.

Apigee

ApigeeX, which is part of Google's Apigee API management platform, is not built on Envoy. Instead, it has its own proprietary architecture and components for handling API management functionalities.

Here's a brief overview of ApigeeX's architecture and components:

- **Proxy servers:** ApigeeX utilizes its own proxy servers to handle API traffic. These proxy servers are responsible for routing requests, applying policies (e.g., security, rate limiting, transformation), logging, and monitoring API traffic.
- **Management server:** The management server in Apigee Edge is responsible for managing the overall API management platform. It handles tasks such as API configuration management, user authentication and authorization, analytics collection, and reporting.
- **Developer portal:** Apigee Edge includes a Developer Portal component, which provides a self-service portal for developers to discover APIs, access documentation, generate API keys, and manage their applications consuming the APIs.
- **Analytics and monitoring:** Apigee Edge offers comprehensive analytics and monitoring capabilities to track API usage, performance metrics, error rates, and other key metrics. It includes dashboards, reports, and alerts for monitoring API health and identifying trends.
- **API gateway:** The API Gateway component in ApigeeX acts as a front-end proxy for APIs, handling incoming requests from clients, enforcing policies, and routing requests to backend services. It provides security features like OAuth authentication, API key management, and threat protection.
- **Policy management:** ApigeeX allows administrators to define and manage policies that govern API behavior and security. These policies can include quota enforcement, traffic management, caching, transformation, message validation, and more.

While Envoy is a popular open-source proxy server commonly used in API management and microservices architectures, ApigeeX uses its own technology stack and infrastructure to provide API management capabilities. It's important to note that ApigeeX is a fully managed API management platform offered as a service by Google Cloud, and its underlying architecture is designed and optimized for scalability, security, and performance in handling API traffic and management tasks within GCP. Future proofing against scale and growing cost of managing Apigee long-term should be carefully considered.

Future-Proof Your Organization with Gloo Gateway

In this whitepaper we've explored the challenges and opportunities presented by the proliferation of microservices in modern software development. We have delved into the transition from traditional API management tools like Apigee to cloud-native solutions and emphasized the advantages of Solo.io's Gloo Gateway, built on Envoy proxy technology, as a forward-looking alternative. We've discussed the complexities of managing APIs in microservices architectures, the evolution of API management tools like Envoy, and the specific challenges and benefits associated with each. We also highlighted the capabilities of Gloo Gateway in addressing these challenges and providing a comprehensive API management solution for cloud-native environments.

As organizations navigate the complexities of managing APIs in the era of microservices, the choice of API management tools becomes critical in ensuring scalability, agility, security, and future-proofing. Traditional solutions like Apigee have served their purpose but may face limitations in cloud-native environments characterized by dynamic service discovery, scalability demands, and diverse technologies.

Solo.io's Gloo Gateway emerges as a compelling solution, leveraging Envoy proxy technology to provide dynamic configuration, seamless integration with service meshes, extensive plugin support, built-in observability, and robust API gateway capabilities. By adopting Gloo Gateway, organizations can overcome the challenges of API management in cloud-native environments and position themselves for future success in managing complex API ecosystems efficiently and effectively.