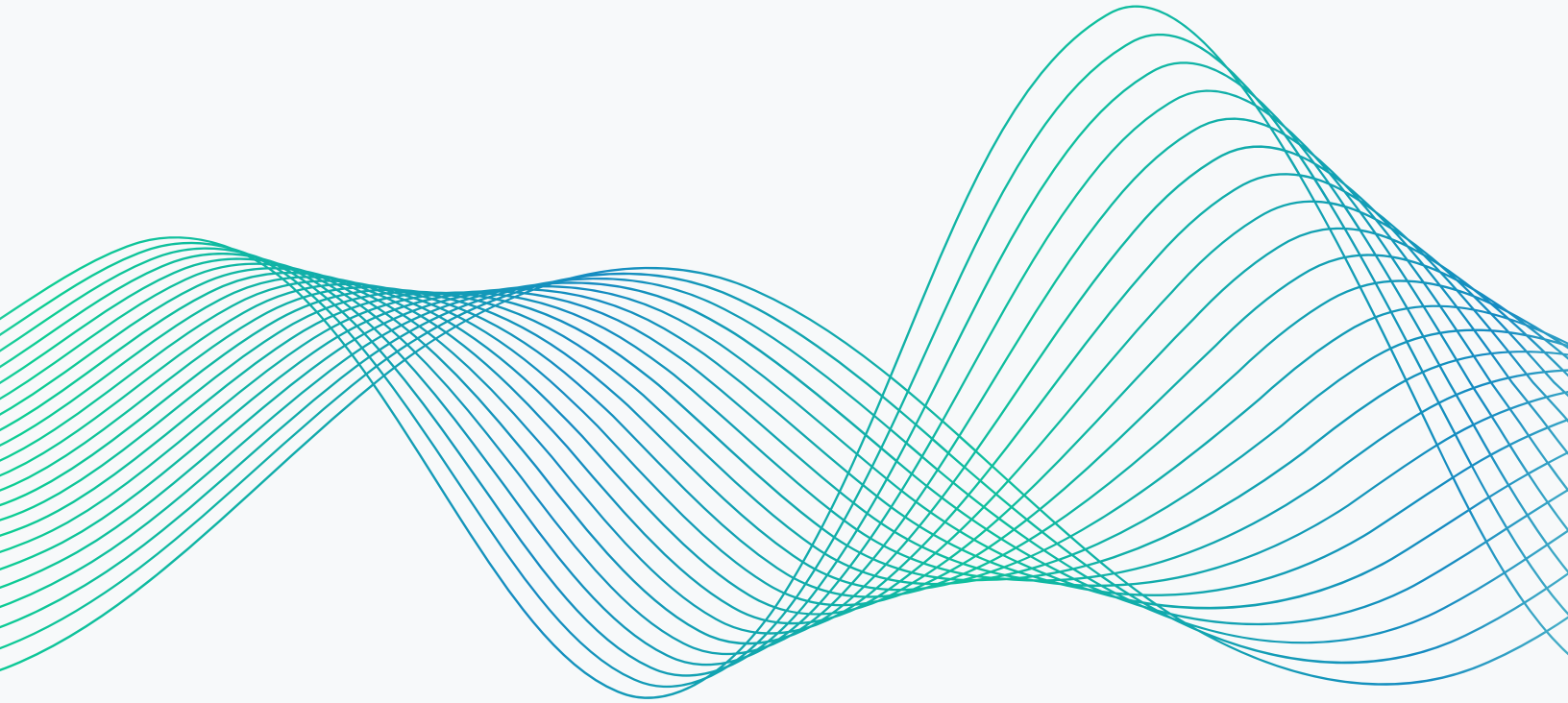
WHITEPAPER

The Value of an Enhanced Istio Service Mesh in Large Environments

See how Gloo Mesh operates in modern application environments at scale



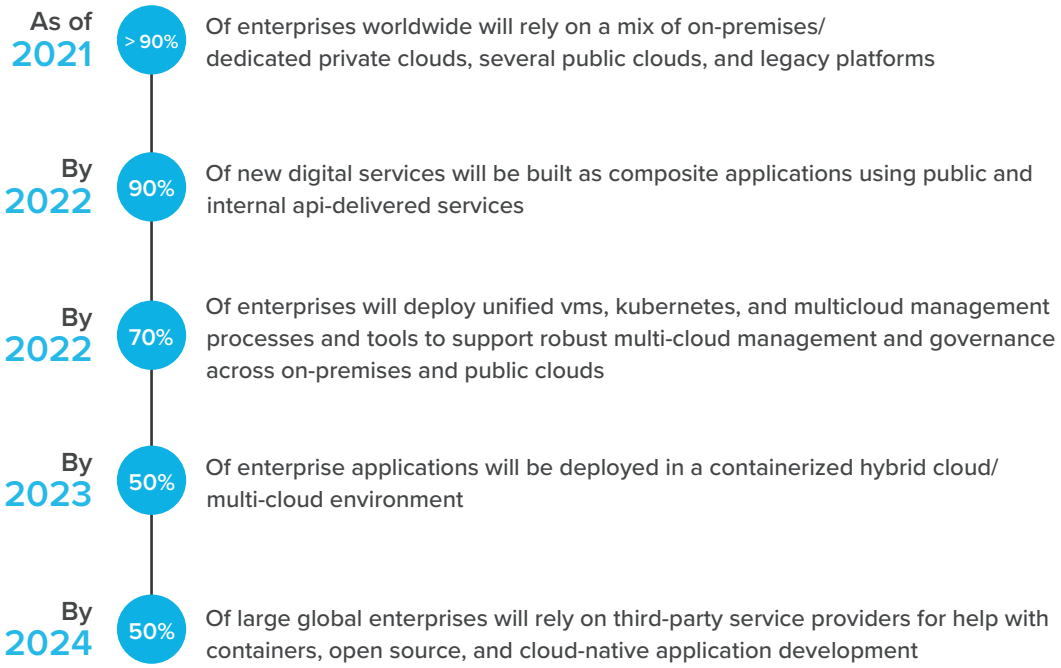
solo.io

Context

You might be starting to plan a digital transformation initiative right now. Perhaps you're already well underway. Sooner or later in these big application modernization projects, you're bound to discover the need for a service mesh to manage secure connectivity between your distributed microservices. When you do realize this need, you're likely to join the vast majority of the market and choose Istio as your foundation. This paper is going to focus on why Solo.io's Gloo Mesh is absolutely the best choice for Istio service management for large, complex application operating environments.

Industry Trends Push for Service Mesh

First, let's start with some current market predictions, based on data and insights from an industry analyst firm in the report *IDC FutureScape: Worldwide Cloud 2020 Predictions:*



As we noted up front, your organization probably has multiple modernization and cloud initiatives going around these trends now, and if not now then very soon. Here are the implications. First, you will need to connect on-premises, hybrid, and multi-cloud environments. Second, you will need to interconnect large numbers of modern Kubernetes containerized microservices applications. Third, you will need to connect your existing applications running on bare metal, virtual machines (VMs), or container-based applications, to bridge between legacy monoliths and more modern application architectures. Legacy approaches to load balancing or API management won't solve these challenges.

Istio offers a programmable way to create and manage a service mesh that runs natively co-located within Kubernetes -orchestrated containers in hybrid- and multi- cloud environments.

A service mesh can solve many of these challenges and requirements. For a hypothetical example, behind the scenes in your bank there could be microservices or distributed applications, separated so they can be developed and run independently, often with Kubernetes as a container orchestrator to manage them. A service mesh interconnects these many microservices so they can talk to one another. Your bank may have massive numbers of different clusters and pods to handle customer login, money transfers, statements, and databases behind them. Your bank probably also has redundant copies of these apps and databases to handle scale and business continuity with failover in case of problems.

The open source project Istio is the most popular service mesh for Kubernetes and cloud environments, with a large and active community backing it, as it was designed to be modern and native, not a retrofit of older, legacy application networking software. This offers a programmable way to create and manage a service mesh that runs natively co-located within Kubernetes -orchestrated containers (and even virtual machines) in hybrid- and multi-cloud environments. Istio abstracts management of the connectivity from the applications themselves, making it easier for developers and operators. Solo.io's Gloo Mesh is a service mesh building on the strengths of Istio.

Istio basics

We need to establish some context around Istio service mesh management. As applications are decomposed from monoliths, all of the resulting microservices need new tools to address the connectivity challenges that arise in handling distributed services. Modern applications are often composed of tens, hundreds, or more microservices that run in containers distributed on-premises and in the cloud. A service mesh defines both the control plane (to configure desired service connectivity and behavior) and the data plane (to direct traffic and enforce security rules). Common challenges include:

- Correctly routing traffic between all of the distributed application services
- Handling issues and errors with retries, timeouts, circuit-breakers, and failover
- Securing connections, including authentication, authorization, and encryption
- Observability and troubleshooting of connections and traffic between services for traffic

Without a service mesh, all of these capabilities would need to be built directly into all the various microservices — a complex and unsustainable task!

The biggest drawback to this alternative is that policies around network control and observability would then be baked into the application code and make it very difficult to change. A big reason for using cloud infrastructure and microservices is to increase the speed of changes to these systems. If these application networking policies are not properly decoupled, it will slow down changes to the system.

How is Istio used?

An Istio service mesh provides essential capabilities around traffic management, security, observability, and reliability to ensure communications between microservices run as expected. Istio has had years to mature into a robust solution for enterprise environments, but also continues to develop many new innovations with releases on a predictable quarterly cadence.

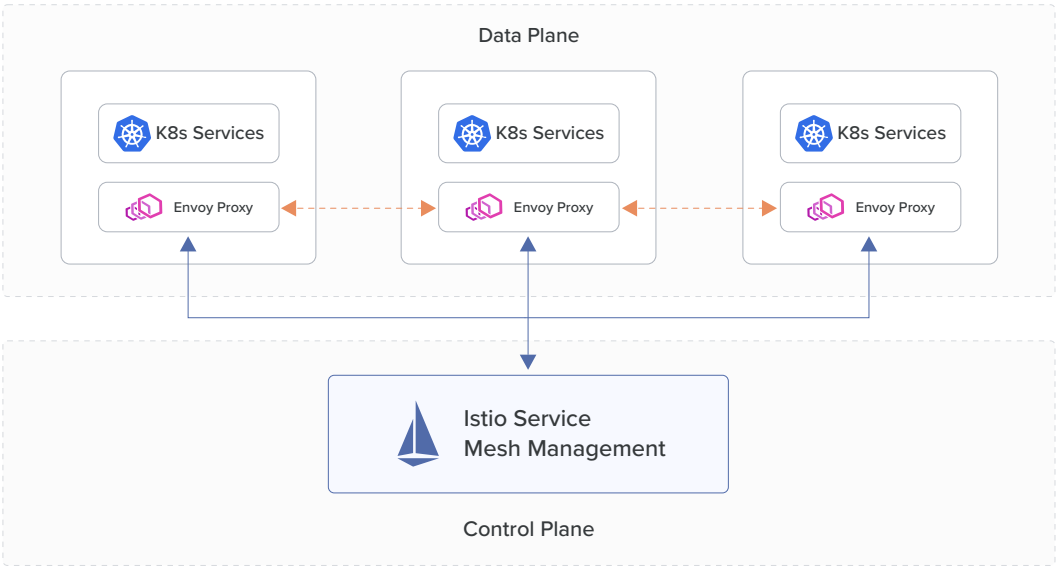
If you are wondering "what is Istio going to look like in my environment?", there are a few ways to think about its functions. Istio lets you define the resources that make up your microservices and applications, and lets you configure rules to securely route OSI layer 4 (L-4) and layer 7 (L-7) traffic between them, including TCP, HTTP, and gRPC protocols. Identifying which resources make up your microservices and applications is the first step in establishing the desired connectivity and routes between them. Istio leverages the open source Envoy Proxy (a graduated project in the Cloud Native Computing Foundation – CNCF) to handle the control plane activities like connections and security. In effect, the Istio control plane acts as a configuration and management layer to give instructions to Envoy Proxies in the data plane. The Envoy Proxies are usually deployed as sidecars within the Kubernetes clusters that support the microservices applications, while Istio itself is usually run in a separate cluster.

You can define basic routing behavior and load balancing, increase reliability with retries, timeouts, and failover, as well as more advanced behavior like rate limiting, quotas, and transforms. Traffic shaping features in Istio enable you to further manage exactly how microservices interconnect and can support canary and A/B testing by splitting traffic, enabling smoother rollout of new application updates with less risk. Fine-grained traffic management between Kubernetes microservices and other applications is the fundamental function of a service mesh.

For security, Istio provides for mutual Transport Layer Security (mTLS) encryption, access controls like authentication, authorization, and vulnerability scanning. Security is essential to protect sensitive information transmitted between microservices on your service mesh. Most customers aim to adopt a "zero-trust" security model which identifies connection requests and denies any/all unvalidated or unsecured connections, from both internal or external sources.

Istio gives you observability of your application communications with telemetry, tracing, and logging for audits. Istio offers compatibility with other open source tools like Prometheus, Grafana, and Jaeger. It also collects and aggregates traffic flow metrics and errors from across all points in the service mesh. This observability is critical to discover, analyze, and address issues around connectivity, performance, security, and other real-world behavior of your service mesh.

Istio architecture



How to implement Istio

As an open source project, Istio can be downloaded directly from community-led repositories on GitHub or sourced and licensed from a commercial provider like Solo.io. The Istio control plane is deployed in Kubernetes clusters with open source Envoy Proxy gateways and sidecars to operate the data plane itself, and allows you to configure and enforce your policies for both North-South and East-West traffic.

Customers deploy and install Istio using Helm charts and/or YAML files in Kubernetes to both push the software and configure it. The Istio control plane is used to further manage the configurations, set policies, and perform updates. Many choose to use the Istio command line “istioctl” to programmatically define and implement configurations and changes. Once Istio is deployed and configured, the next step is to define the services in the mesh. Envoy Proxies are usually set up as sidecars in each of the Kubernetes application clusters.

You can implement and manage Istio yourself, but you should think about what Istio is going to need in terms of investment. Certainly Istio will require a lot of administrative effort to self-support and adapt to enterprise requirements, or you can choose a more comprehensive Istio management product such as Gloo Mesh, which comes with enterprise production support. If you want to make it easier for your API producing and consuming developers, an Istio-native developer portal enables GitOps and CI/CD methodologies.

The main benefit of Istio is enabling modern applications such as containerized microservices in hybrid- and multi-cloud environments to connect safely, reliably, and securely.

Benefits of Istio

The main benefit of Istio is enabling modern applications such as containerized microservices in hybrid- and multi-cloud environments to connect safely, reliably, and securely.

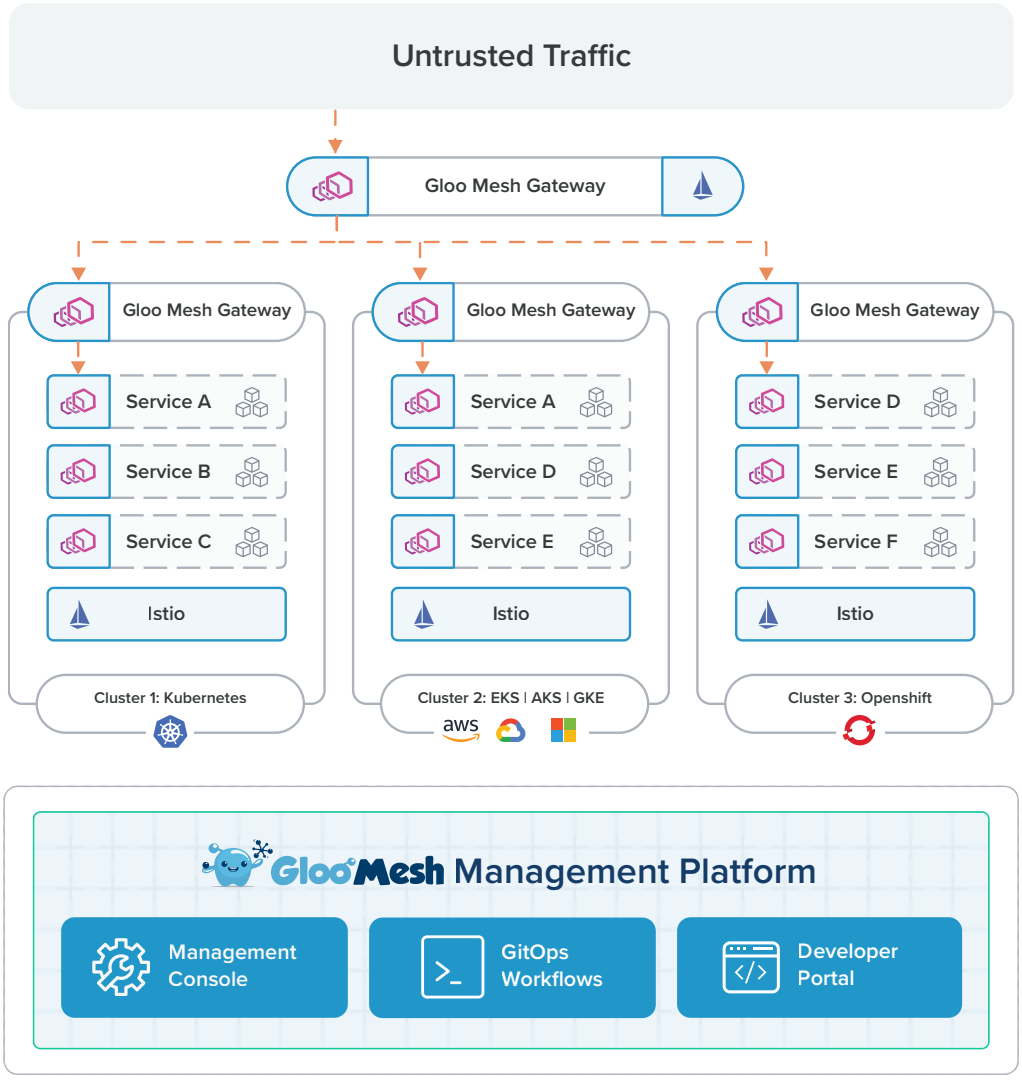
Without an Istio service mesh, tools to manage your desired behavior around connectivity and security would have to be implemented directly in each application – which would not be very efficient, consistent, or scalable. Istio abstracts the control plane and data plane from the applications and physical infrastructure, making it much easier to manage, secure, and observe your service mesh. Istio also lets you split traffic to support A/B testing, blue/green, canary deployments of applications for GitOps and CI/CD style application development.

Older offerings around API gateways simply don't achieve all the capabilities of Istio, since they were developed without being Kubernetes- and cloud-native. While some vendors have tried to adapt their products, they have built-in architectural limitations that will make it much harder for teams to achieve their objectives.

From a business standpoint, adopting an Istio service mesh means you will have reduced risk, increased security, and easier management of the connectivity between Kubernetes-based and legacy applications. Istio even helps with application modernization and "migration to cloud" initiatives by providing a way to direct traffic between new microservices and application updates.

What is Solo.io's Gloo Mesh?

Gloo Mesh provides a Kubernetes-native management plane with an enhanced Istio distribution, and improvements to Envoy Proxy delivering API gateway and service mesh capabilities together. Key features include Istio lifecycle management, multi-cluster security/traffic management, global routing and failover, policy management, global service discovery, multi-tenancy and role-based access controls, extensibility, and multi-cluster observability and operations. The diagram below shows how Gloo Mesh might be deployed in a modest environment. Of course you can scale tens of thousands of services for a large environment, it's just that would be hard to show in our diagram. We'll go into how Gloo Mesh compares to open source Istio in detail a bit later.



At Solo.io, we have had hundreds of in-depth conversations with large organizations and what we consistently hear is where there are feature gaps in open source software or a vendor's offering, it becomes your responsibility to fill them.

Why do you need Solo.io's Gloo Mesh?

Getting value from your service mesh

Be warned: not all service meshes are created equal, even if some start from the same open source. Since the connections between users and applications and operating environments are essential for distributed applications, it's clear that both traditional and new IT requirements must be met. What works well enough for a single cluster in a test environment may not scale efficiently or be manageable in large production environments. To succeed, you will need your solution to meet the following five design principles (which overlap a bit):

1. Secure – You need a zero trust model and end-to-end controls to meet best practices and strict regulations, or you will face increased effort and risk. The bigger the environment, the harder it is to secure comprehensively and consistently.

2. Reliable – You need robustness for mission-critical workloads, with centralized control, or you will face increased risk of errors and outages. Troubleshooting and ensuring high service availability becomes tricky as you go from hundreds to thousands of microservices.

3. Unified – You need consistent observability and management at scale for your choice of environments and policies, or you will face increased effort and risk. Managing per application, per cluster, or per mesh can become unsustainable as you grow.

4. Simplified – You need reduced complexity and expert advice so you can innovate and modernize faster, or you will face increased effort and risk, which both can compound exponentially in big environments.

5. Comprehensive – You need complete solutions, configurable and extensible for all your specific needs, or you will face increased effort. If a technology only works for part of your environment, or in one cloud, or only for Kubernetes, you may find yourself running redundant tools and doubling your overhead.

These principles are perhaps not surprising, but few options on the market can meet these needs well for large application environments. At Solo.io, we have had hundreds of in-depth conversations with large organizations and what we consistently hear is where there are feature gaps in open source software or a vendor's offering, it becomes your responsibility to fill them. This means more effort developing and maintaining custom code to make it work. We get asked a lot about build vs buy, and while we love to help you customize and create your own approaches, why not save yourself all the extra effort and risk? Do-it-yourself solutions carry a real — if soft — cost (staff salaries), an opportunity cost (higher value things you could be doing instead), delays (writing and maintaining custom code takes time), and increased risk (compatibility issues, security breaches, and service outages, and initiative failures.)

Solo.io advantages for getting value from your service mesh

For example, Vonage said, “Our first set of challenges came when it was time to upgrade to a newer version of Envoy Proxy. Some of the APIs we were using to build our bridge to the authentication service had changed, upgrading took days and we didn’t have a deep bench of C++ expertise just to support Envoy — so we started to look for alternatives.”

Only Solo.io can meet all these requirements, with unique capabilities that should quickly eliminate other alternatives from your consideration. More importantly, this is a durable advantage for you, because we got there by starting from the right platform with the right team. DIY with open source may look easy, but there are many hidden costs and risks, and they will be forever issues, not a one-time effort.

Here are some examples of features to prove the point, aligned to the six principles above:

1. SECURE

Basic open source Istio includes some basic security features like:

- Transport Layer Security (TLS/mTLS) to provide end-to-end encryption to protect data in motion between end points.
- Vulnerability scanning which finds, addresses, and alerts on weaknesses in the system.
- Secrets integration with Kubernetes to manage sensitive credentials like passwords, tokens, and keys.

Gloo Mesh goes much further and also offers these features which are essential in large environments:

- Multi-tenancy and isolation that lets service meshes share resources securely.
- Federated trust domains to safely authenticate users and applications across environments.
- Federated role-based access control and delegation to grant permissions to users appropriate to their responsibility and applies them consistently everywhere.
- Safe handling of signing certificate and root rotation to manage and execute SSL certificates from a centralized platform.
- FIPS (140-2) compliant builds validated to meet strict security standards.
- A secure configuration model for cluster relay safely shares configurations across the system.
- External Authentication that integrates with API keys, JSON web tokens (JWT), lightweight directory access protocol (LDAP), OAuth, OpenID Connect (OIDC), and custom services.

- OIDC/Oauth 2.0 flows to manage authentication of users and applications.
- Built-in web application firewall (WAF) based on open source ModSecurity to screen traffic for threats and stop attacks.
- Data loss prevention (DLP) to monitor for data breaches or exfiltration to prevent data loss and data leaks.
- Use of Open Policy Agent (OPA) for authorization which defines service API policies as code.

2. RELIABLE

Basic open source Istio includes:

- Retries, circuit breaker, timeouts to handle exceptions and issues in connections gracefully.
- No-interruption updates to roll out new configurations and policies without requiring restarts or pausing operations.
- Health checks to confirm that the system is operating as expected.
- Configuration validation that makes sure that the system is deployed and defined correctly.

Gloo Mesh goes much further and also offers these features which are essential in large environments:

- Multi-cluster dynamic routing to steer connections on-the-fly to available resources across clusters as needed.
- Priority failover routing that defines in which order alternate resources should receive re-directed traffic in the event of a service outage.
- Published SLAs which provide assurance that issues are responded to in a timely manner.
- Dynamic scaling to thousands of nodes which robustly manages regular and unexpected variations and spikes in workloads.
- Simplified global service naming so you can use consistent naming across all clusters.
- Advanced rate limiting to define custom policies to handle more complex situations.

Basic open source Istio isn't easy, and as with all open source, you'll be reliant solely on the kindness of strangers and your own efforts to maintain software and troubleshoot issues.

3. UNIFIED

Basic open source Istio includes:

- Distributed tracing (integration with Jaeger) which facilitates root cause analysis of issues across the system.
- Multi-cluster security policies implemented consistently across all environments to avoid exposure or risk of errors.
- Multi-version compatibility that enables running different versions of Istio together so you can upgrade at will.

Gloo Mesh goes much further and also offers these features which are essential in large environments:

- Global service discovery which finds, defines, and maps resources (applications/microservices) that can be targets for connections.
- Federated multi-cluster operations and policies to manage, push configurations, and monitor activity across clusters and even hybrid and multi-cloud deployments.
- Multi-cluster observability (including Prometheus and Grafana) that collects system metrics for observability to monitor and troubleshoot, and auditing for investigation and displays system metrics in user-friendly graphs and enables building custom dashboards.
- Cross-origin resource sharing (CORS) to set policies and pre-verify which origins are allowed to connect to specified resources.
- An admin dashboard GUI with multi-cluster views which gives centralized observability and control of the whole system.
- Gloo Developer Portal (API mgmt) which enables publishing, sharing, GitOps calling, and monetization of defined APIs.

4. SIMPLIFIED

Basic open source Istio isn't easy, and as with all open source, you'll be reliant solely on the kindness of strangers and your own efforts to maintain software and troubleshoot issues.

Gloo Mesh goes much further and also offers these features which are essential in large environments:

- Simplified API that makes it easier to configure and use Istio (and Envoy Proxy).
- Long-term version support that covers releases of Istio and Envoy for at least a year so you can upgrade on your schedule.
- N-4 version patching & back-porting of fixes for bugs and security issues in current and four previous releases of Istio and Envoy.
- Expert help on Slack for fast response to all your questions by an active public community and Solo engineers worldwide.
- Enterprise support which helps quickly resolve issues in production environments via Slack, email, and phone.

5. COMPREHENSIVE

Basic open source Istio includes:

- The ability to shape, shift, and transform traffic to define exactly how you want requests to be processed and presented, and connect to diverse protocols.
- Virtual machines (VMs) support that enables connections to VMs alongside containers and serverless upstream resources.
- Your choice of cloud and on-premises environments to operate anywhere you choose to operate your applications.
- GitOps workflows to manage applications and operations on-demand.
- WebAssembly (Wasm) which provides the ability to define extensible custom filters for security and control.

Gloo Mesh goes much further and also offers these features which are essential in large environments:

- Global service routing which directs application connections across any environment for choice and reliability.
- Locality-aware load balancing that manages routing of workloads across distributed resources to achieve best performance and results.
- Support for ARM to operate efficiently on high performance processors for compute.
- Serverless functions integration that enables connections to AWS Lambda alongside containers and other upstream resources.
- Simple object access protocol (SOAP) transforms to tie in XML messaging protocols for legacy applications.
- Multi-cluster Wasm to manage your customizations consistently everywhere.
- Announced plans to integrate GraphQL with Gloo Mesh.

If you take away nothing else from this paper, here's the main point: You don't have to reinvent all of those features above. You can find success much faster with much less effort and much less risk if you choose Solo.io's Gloo Mesh for Istio in your large environments. Some vendors have started from the wrong foundation (something not originally built for Kubernetes or cloud) or picked the wrong project to back (without market adoption or momentum to keep it going).

Gloo Portal can accelerate developer onboarding with self-service documentation, and self-service sign up.

What is Gloo Portal?

Gloo Portal is an extension to Gloo Mesh to catalog, publish, and securely share APIs via a self-service portal, which is extremely helpful for large environments with lots of independent application developer teams. It's CRD-driven and works flawlessly with existing GitOps and CI/CD processes. Gloo Portal can accelerate developer onboarding with self-service documentation, and self-service sign up. You can manage gRPC APIs and REST APIs in the same developer portal, and upload existing OpenAPI and proto documents to build the catalog. Gloo Portal helps you communicate authentication/authorization instructions, usage plans, and policies.

Helping customers with expert advice and best practices

There's also an intangible advantage to Solo.io, and that's our team. Few people have the necessary knowledge of Istio service mesh, while all Solo.io's Istio contributors and experts are available to tailor solutions for your large environments. We have the best and the brightest standing ready to help you whenever you need us, including for design advice, deployment help, and enterprise Istio support. Our leadership team has come from Google, VMware, AWS, IBM, RedHat, EMC, Intel, Apigee, and Gartner, but they've all recognized the special opportunity Solo.io has to solve critical needs in the market. We are active in the Istio open source community and contribute back regularly. We collectively have over 200 patents and more coming. We've literally written the books on these technologies (Lin Sun's "Istio Explained" and Christian Posta's "Istio in Action".) We deliver regularly with a track record of innovation. Our products aren't an under-funded "hedge-our-bets" play in a huge company's portfolio, we are focused on service mesh.

"Collaborating with the team at Solo.io has been great. They are very responsive in helping us with our Gloo environment, brainstorming ideas on how to solve our issues and responding to the questions we've had on Kubernetes like load balancing and how the ecosystem of tools works together. They are truly invested in our success," said Jonathan Lane, senior manager, software engineering, API platform at Vonage.

Ask an expert and you can save yourself a lot of time and effort and help you in getting value from your modern application initiatives. We can help you succeed with service mesh in large environments in the following areas. Reach out to our team today!

Learn more about Gloo Mesh:

- Request a Gloo Mesh trial license
- Get in touch with us for a Gloo Mesh demo

ARCHITECTURE & DESIGN

Got a question about design, best practices? We have helped hundreds of enterprise customers plan their deployments to meet complex requirements.

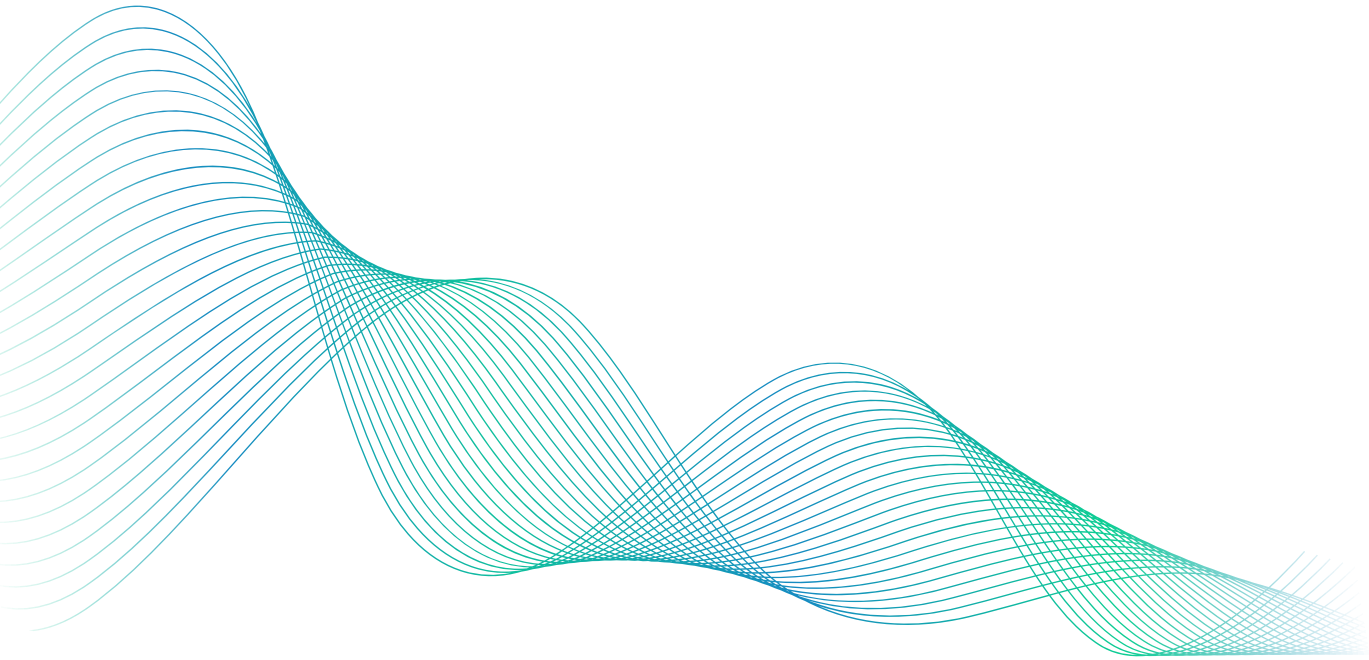
SECURING YOUR ENVIRONMENT

We can explain how to implement a zero-trust model with authentication, encryption, firewall, role-based administration, custom filters, and FIPS compliance.

OPERATING IN PRODUCTION AT SCALE

Let us show you how to manage, observe, and troubleshoot connectivity for thousands of microservices in hybrid, multi-cluster, and multi-mesh environments.

Talk to an Expert



solo.io
contact us

About Solo.io

Solo.io, the modern service connectivity company, delivers API infrastructure from the edge to service mesh, helping enterprises adopt, secure, and operate innovative cloud native technologies. APIs drive microservices and cloud native technologies, forming the foundation for developers, partners and customers to interact with application services quickly, effectively, and securely. Solo.io brings developer and operations tooling to manage and federate security and traffic control and tie together the integration points to enable and observe the application network.