solo.io

eBook

Driving Business Value with Istio

How a service mesh can help your organization simplify the adoption of a distributed architecture

Table of Contents

Introduction

Whether we're talking to our customers, community members, or other experts in the field, what we hear consistently is that their major hurdles with application modernization are security, resiliency, and observability.

Organizations are at a major point of transition, moving from monolithic applications to microservices. That's with good reason – microservices allow for greater agility and faster delivery and time-to-deployment, increasing customer satisfaction and streamlining business operations.

The challenge with microservices, when compared to traditional application servers, is that security, resiliency, and observability are down to each one of the individual teams managing the different microservices – there's no default uniformity or consistency.

A service mesh is an infrastructure layer that aids in communication between services or microservices using a proxy. **Istio is a service mesh that helps organizations address and simplify many of the common issues associated with a distributed architecture.**

In the following sections, we'll walk through what those challenges are, how a service mesh can help, and why Istio in particular is the right solution.

The Common Challenges of Managing Microservices

The Impact on Security

With microservices highly distributed across a multi-cloud ecosystem, there is a multiplication of potential attack vectors, and the threat of cyberattacks is greater than ever. Security incidents risk organizations losing revenue and reputations, which can also cost customers and prospects.

93%

93% of respondents reported at least **one security incident** in their Kubernetes environment in the past 12 months.[1]

31%

31% of respondents say they have experienced **revenue or customer loss** due to a security incident over the last 12 months.[2]

$4.24M

The **average cost of a data breach** is $4.24 million.[3]

1 https://www.redhat.com/rhdc/managed-files/cl-state-of-kubernetes-security-report-2022-ebook-f31209-202205-en.pdf
2 https://www.redhat.com/rhdc/managed-files/cl-state-of-kubernetes-security-report-2022-ebook-f31209-202205-en.pdf
3 https://qz.com/2039599/why-the-cost-of-getting-hacked-is-higher-than-ever

The Impact on resiliency

Service downtime, like an infrastructure or service failure, directly impacts revenue. Any issues with service delivery affect customer satisfaction and trust.

$1M Per Hour

For Fortune 1000 companies, downtime could cost as much as $1 million per hour[4]

$1M Per Year

A typical mid-sized company spends $1 million per year on incidents[6]

$5M Per Hour

Downtime in high risk industries (banking, finance, government, healthcare, manufacturing, and media/communications) can cost as much as $5 million per hour[5]

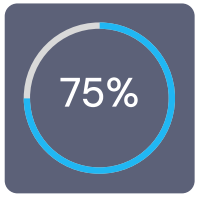
Business Impact of Downtime

lower customer satisfaction, loss of revenue, reputation loss, loss of customers[7]

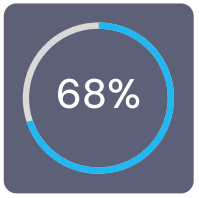
4 https://www.atlassian.com/incident-management/kpis/cost-of-downtime
5 https://trilio.io/resources/cost-of-downtime/
6 https://www.atlassian.com/incident-management/kpis/cost-of-downtime
7 https://www.atlassian.com/incident-management/kpis/cost-of-downtime

The Impact on Observability

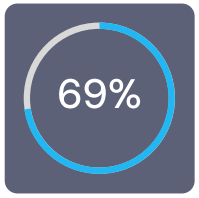
With such huge costs at stake from a security and resiliency perspective, organizations need to be able to detect and plan for these problems. Your competitors are – and it shows in their ability to innovate.[8]



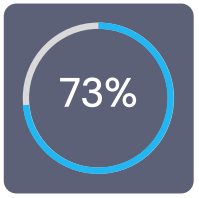
Report an improvement in problem detection time



Say they've seen an improvement in development times



Better mean time to resolution for unplanned downtime or performance degradation



Report an improvement in deployment times

Annual cost of downtime associated with business-critical internally developed apps:

Beginners: $23.8 million

Leaders: $2.5 million

8 https://www.splunk.com/en_us/pdfs/gated/research/state-of-observability-2022.pdf

Understanding the Service Mesh Architecture

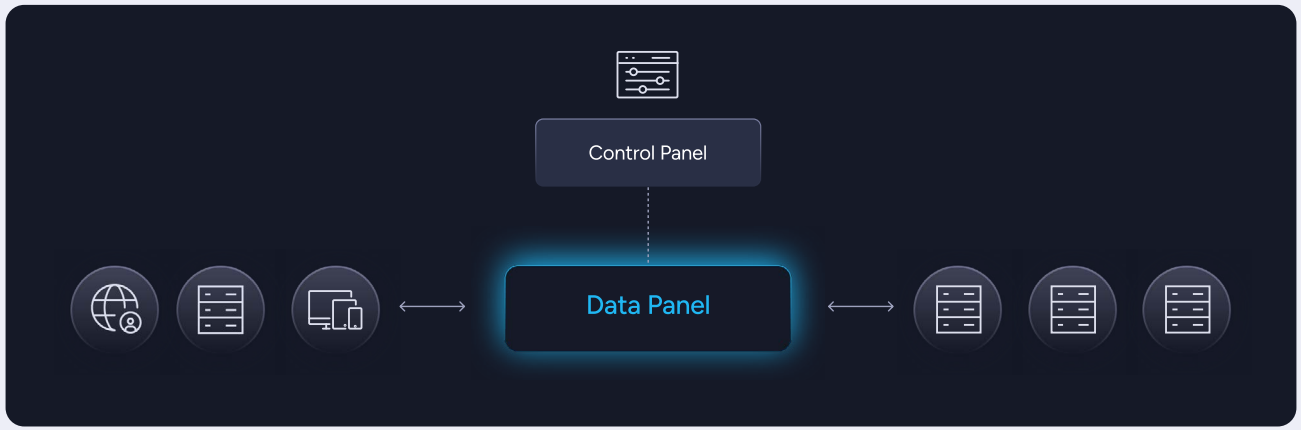
A control plane is where a user describes what they want to happen and defines the policies they want to enforce, and the data plane is where those policies are actually realized, enacted, and enforced. They communicate with each other.

They're important because users can **define policies and configuration around security, resiliency, and observability in a single place**, and deploy them through the control plane, and the control plane can then implement those directly in the data plane without requiring any changes or any buy-in from any of the existing services connected to the data plane.

Istio follows the control plane and data plane architecture pattern. With Istio, the control plane reads the user-defined configuration (Kubernetes Custom Resources), and distributes it to the data plane where the policies are enforced.

Service meshes like Istio help users gain more insight and control of distributed application behavior. They provide essential capabilities to application developers, including service

communication-layer security, service discovery, client-side load balancing, timeouts, retries, and circuit breaking.



Istio Use Cases

Here are a few scenarios – of many – where Istio can be especially valuable, saving your company time and money by automating important tasks while mitigating risk.

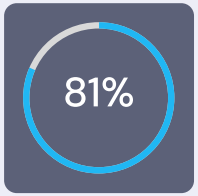
Automatic Security

Networking, and enforcing mTLS in particular, is a primary use case for Istio.

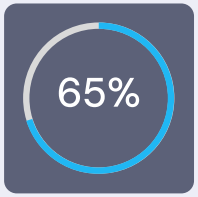
Encryption of the data in transit is the primary benefit of TLS, while mTLS adds the ability to validate the identity of both client and service. Normally, network traffic in Kubernetes clusters is unencrypted and anyone that can get access to the network can observe, opening organizations up to security risks.

That's why zero trust, where the target state is that there are no implicit security assumptions, is so valuable. All workloads authenticate and are authorized to access other services.

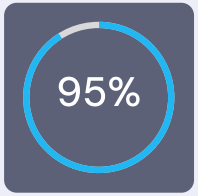
We know zero trust is critical, but it's challenging for companies to internally implement on their own:



81% of companies experienced a certificate-related outage in the past two years.



65% are concerned about the increased workload and risk of outages caused by shorter SSL/TLS certificate lifespans.



Human error was a major contributing factor in 95% of breaches.

Automation makes that implementation much easier, and that's what Istio does for security. With Istio, users get:

Transport Security

Which automatically encrypts communication between all services in transit.

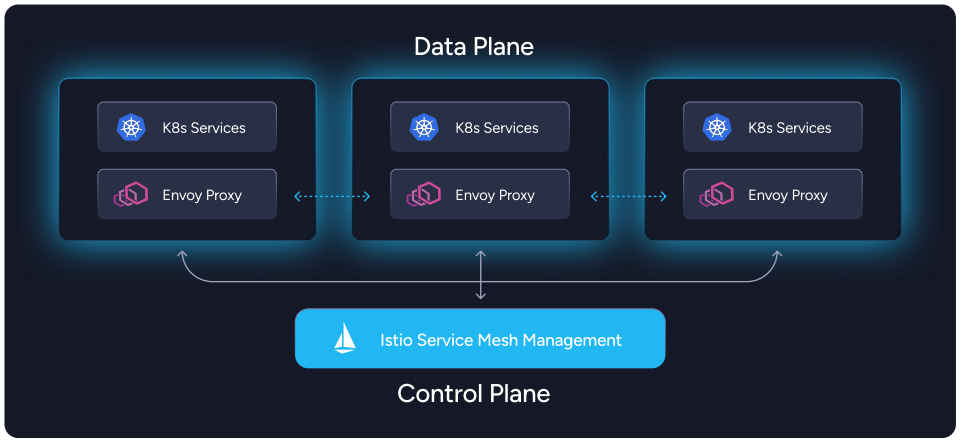
Identity Authorization

Which provides strong identities for both the client and server.

Certificate Management

Where the certificates that are used are automatically refreshed and the lifestyle of those certificates is controlled by Istio itself.

Istio helps eliminate the human failure modes for non-compliance or what's associated with downtime for expired certificates. With Istio, you get instant mTLS.



Allowing for failures

In cloud native environments, containers and services go up and down – and there are going to be failures. Often, those failures are transient and happen for a short period of time, eventually going away. We know that in regard to being resilient:

- **Waiting indefinitely is bad:** services need to be controlled for when other dependent services are unresponsive
- **Trying multiple times is a good practice:** if a service fails once, then it should be set up to try again
- **Allowing for degradation is important:** when services get overwhelmed, or go bad permanently, they need to be taken out of action gracefully

There are common mitigation strategies for these failures:

Implementing timeouts that say to only wait a certain period of time, and then terminate the connectio

Allowing for a certain number of retries in the event of a service interruption

Flipping a circuit breaker to limit the amount of traffic before it overwhelms a service and causes a full outage for all customers

While the Kubernetes orchestration layer is built to keep services running, Istio can help users keep resiliency strategies in place that control how all services communicate with each other, with no code changes or intervention needed from development teams.

Gaining valuable insights

With development teams deploying many different types of services, users need to be able to look at the overall system in a consistent and uniform manner, then take action based on that view. Building a uniform approach across teams means everyone can:

Understand Traffic Patterns

Determine Service Health

Anticipate Outages

Detect Dangerous Activity

Audit Access

Through Istio, there are standardized metrics, tracing, and access logs that serve as a true recording of what went into or out of a given service – no matter what team deployed it or which programming language it was written in. This valuable data can be viewed in out-of-the-box and existing observability dashboards.

That level of observability can **save millions of dollars a year**, allowing users to get ahead of potential downtime issues.

How Gloo Platform Makes Istio Even Better

As an open source project, Istio can be installed by anyone, but as a stand-alone package, management of Istio, along with other Kuberenetes components, can be very difficult and time consuming. When deployed in large environments with multiple teams and clusters, configuring Istio directly can be complicated and error prone.

Solo.io's Gloo Platform is built with Istio at its core – every feature we've built around it has been informed by the enterprise needs of our customers as they adopt Istio. Gloo Mesh is a distribution of the Istio service mesh that is hardened for production support across multiple teams and multicluster hybrid clusters. Gloo Mesh has a simplified configuration model that is catered to application developers and SREs, which is easier to use and less error prone than the basic Istio configuration

With Gloo Mesh, you can unify the configuration, operation, and visibility of service-to-service connectivity across your distributed applications. These apps can run in different virtual machines (VMs) or Kubernetes clusters on premises or in various cloud providers, and even in different service meshes.

Benefit	Gloo Mesh Enterprise	Gloo Mesh Open Source	Community Istio
Upstream-first approach to feature development	✓	✓	✓
Installation, upgrade, and management across clusters and service meshes	✓	✓	✗
Advanced features for security, traffic routing, transformations, observability, and more	✓	✗	✗
End-to-end Istio support and CVE security patching for n-4 versions	✓	✗	✗
Specialty builds for distroless and FIPS compliance	✓	✗	✗
24x7 production support and one-hour Severity 1 SLA	✓	✗	✗
GraphQL and Portal modules to extend functionality	✓	✗	✗
Workspaces for simplified multi-tenancy	✓	✗	✗

The platform we've built around Istio has enriched the service mesh even further, identifying the gaps in Istio and addressing them. Gloo Mesh includes n-4 Istio version support with security patches to address Common Vulnerabilities and Exposures (CVE), as well as special builds to meet regulatory standards such as Federal Information Processing Standards (FIPS). The enterprise features also include multi-tenancy, global failover and routing, observability, and east-west rate limiting and policy enforcement through authorization and authentication plug-ins.

Gloo Mesh is built with the following six principles to improve how you can introduce a service mesh into your cloud native environment with confidence:

Secure

You need a zero trust model and end-to-end controls to implement best practices, comply with strict regulations like FIPS, and reduce the risk of running older versions with security patching.

Unified

You need one centralized tool to manage and observe your application environments and traffic policies at scale.

Simplified

Your developers need a simple, declarative, APIbased method to provide services to your apps without further coding and without needing to understand the complex technologies like Istio and Kubernetes that underlie your environments.

Reliable

You need a robust, enterprise-grade tool with features like priority failover and locality-aware load balancing to manage your service mesh and API gateway for your mission-critical workloads.

Comprehensive

You need a complete solution for north-south ingress and east-west service traffic management across infrastructure resources on-premises and across clouds.

Moden and Open

You need a solution that is designed from the ground up on open source, cloud native best practices and leading projects like Kubernetes and Istio to maximize the portability and scalability of your development processes.

If your organization is concerned about security, resiliency, and observability, Gloo Platform can help you move beyond problem solving and into scaling and innovating.

About Solo.io

Solo.io, the leading application networking company, delivers a service mesh and API platform for Kubernetes, zero trust, and microservices. The components of Gloo Gateway and Gloo Mesh enable enterprise companies to rapidly adopt microservice applications as part of their cloud journey and digital transformation. Solo.io delivers open source solutions, and is a community leader in building the technologies of the future.

contact@solo.io

www.solo.io

Learn more