tetrate

Five Essential Strategies for Deploying Service Mesh on Amazon EKS

A Practitioner's Guide to Maximize Istio and Envoy on EKS

Contents

Introduction

Virtually all organizations are adopting cloud infrastructure, either augmenting their on-prem footprint or wholesale migrating away from on-prem in favor of more flexibility. Indeed many organizations are creating their entire infrastructure in the cloud from day 1! As of the time of this writing, Amazon Web Services (AWS) holds the largest market share amongst cloud providers. When running workloads in the cloud, many organizations are also adopting cloud-native infrastructure, which starts with running workloads in containers and adopting a container orchestration platform. Kubernetes is the most popular open source container orchestration engine for automating deployment, scaling, and management of containerized applications. Amazon Elastic Kubernetes Service (EKS) is a managed Kubernetes service to run Kubernetes. Amazon EKS automatically manages the availability and scalability of the Kubernetes control plane nodes responsible for scheduling containers, managing application availability, storing cluster data, and other key tasks.

EKS is undoubtedly one of the most popular ways to run Kubernetes, but are you getting the most out of it? Specifically, how do you translate small-scale success in limited-scale deployments into at-sale production infrastructure? Not only does running Kubernetes at scale create new security and availability challenges for its operators. Crucially, Kubernetes teams rarely have the luxury of adding new headcount and budget as they scale out. In search for operational leverage is when Kubernetes operators find service mesh as the solution to achieve their security and availability requirements.

Service mesh is a proven architecture popularized recently that gives operators control and observability over Kubernetes traffic. Istio and Envoy have emerged as the de facto standard for implementing a service mesh, including Amazon EKS. With the right service mesh implementation on Amazon EKS, you can take advantage of all the performance, scale, reliability, and availability of AWS infrastructure, as well as integrations with AWS networking and security services. In this whitepaper, we present six service mesh strategies to help you enhance security and availability for your EKS workloads.

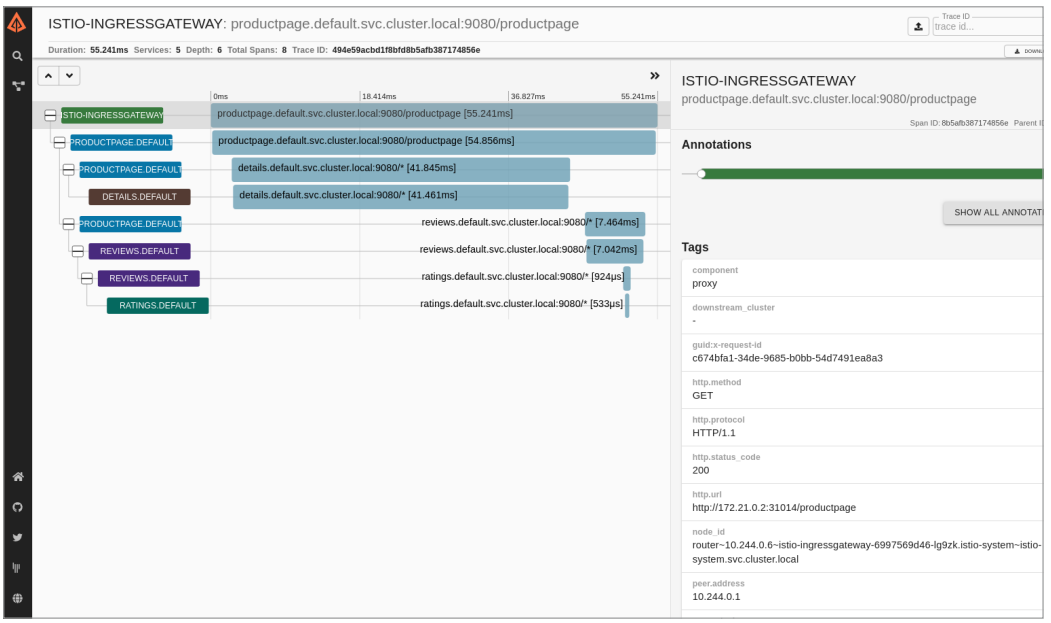
Strategy 1:
Observe Granular Service Traffic

Service Mesh Benefits

Understanding and monitoring the health of all your EKS workloads is challenging because of the sheer volume and heterogeneity of the services. It is not practical to integrate each service with native AWS or 3rd party tooling. Among other things, a service mesh acts as a transparent observability layer on your existing services. In the case of Istio, it generates detailed telemetry for all communication within a service mesh. Best of all—because Istio monitors traffic transparently through Envoy sidecars, application changes are minimal to none for granular observability.

Out of the box, Istio provides three types of telemetry:

- Metrics: Istio generates latency, traffic, error, and saturation metrics in addition to the health of the Istio control plane itself. In addition to summary-level metrics, you can also drill down into detailed metrics at the Envoy proxy level to remediate issues quickly.
- Distributed traces: To understand how an external request flows through services and dependencies, Istio provides distributed traces for troubleshooting issues across services.
- Access togs: Istio captures logs at the per-request level, including source and destination metadata. The log information allows teams to perform analysis at scale to root cause issues.



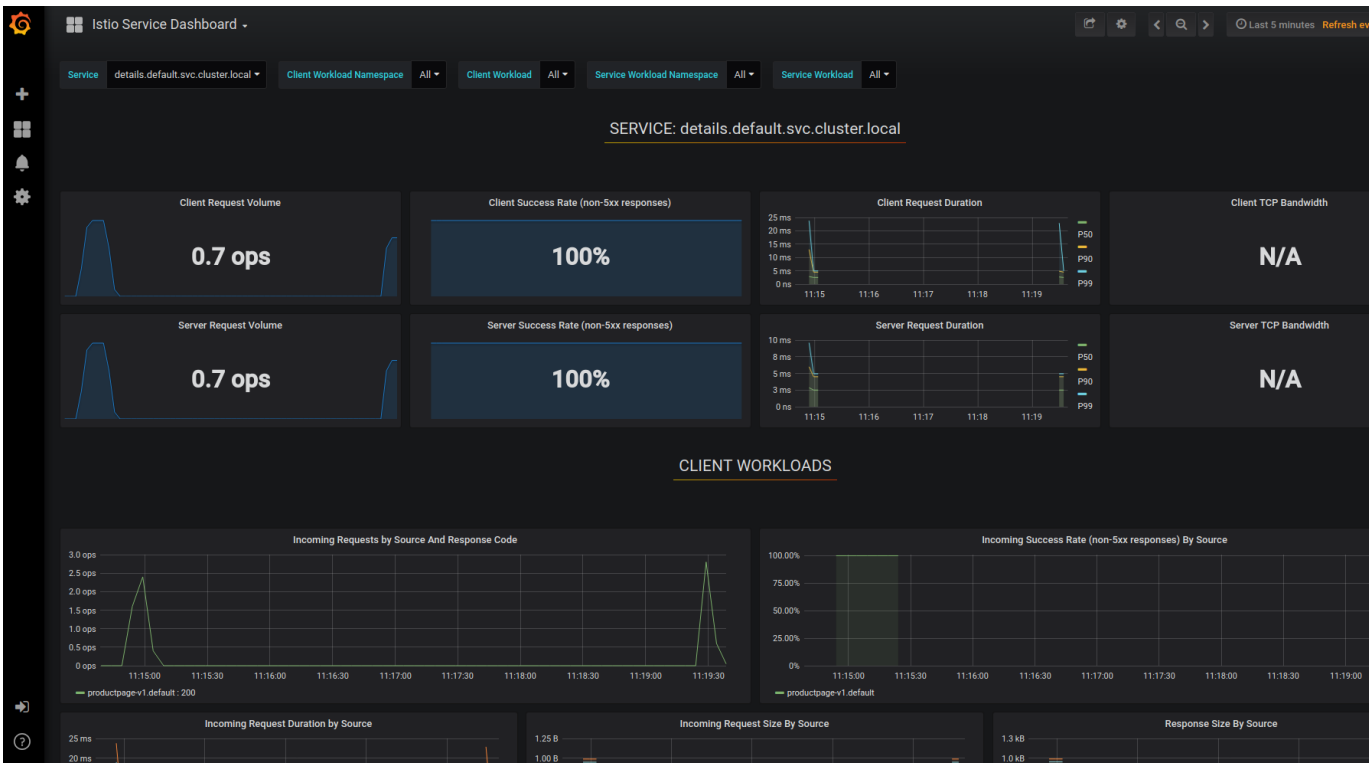
Distributed tracing with Istio

Source: Istio

How to Get Started

Istio is a control plane that has a wide selection of configuration options to turn on and off telemetries with very comprehensive documentation. The standard Istio metrics are exported to Prometheus by default. You need to install and maintain Prometheus before you can monitor these metrics. Prometheus is commonly used with Grafana for data visualization. Istio ships with a default set of Grafana dashboards for monitoring service behaviors based on these metrics.

Teams often build a *management* plane that handles the integration and lifecycle of Prometheus, Grafana, and various other tools in the observability stack. These open source tools are community-supported, but enterprise support options are also available. See the end of this whitepaper for support and out-of-the-box management plane options.



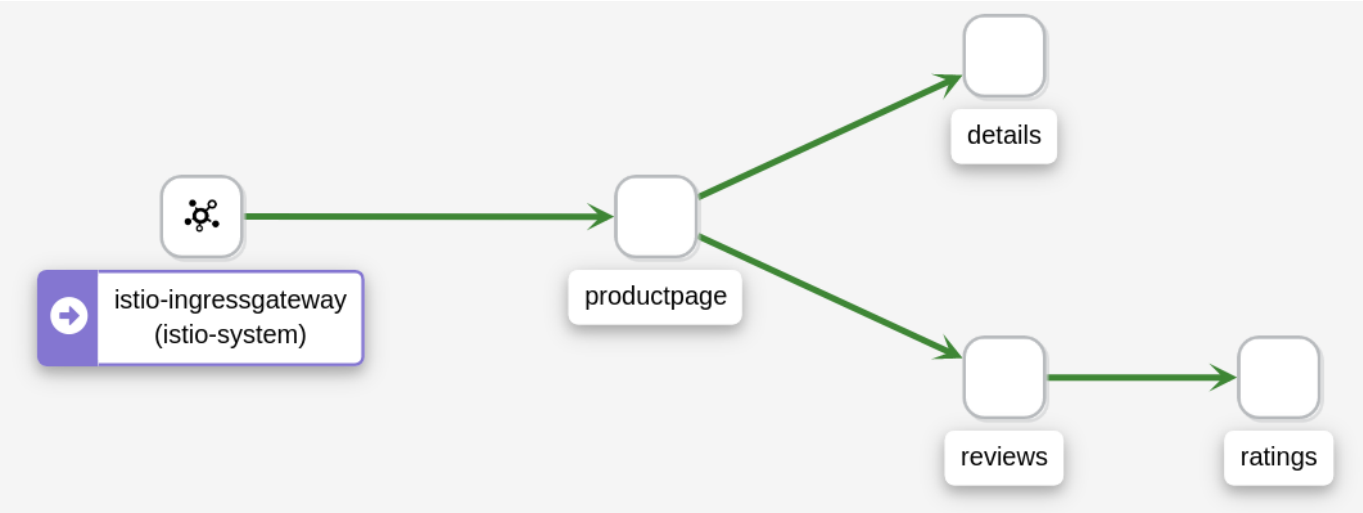
Istio metrics in Grafana

Source: Istio

Strategy 2:
Understand Service Topology

Service Mesh Benefits

As you add more services, understanding how services are interconnected can help you identify potential performance bottlenecks early. Taking the concept of Istio metrics one step further, one can stitch together a global view of service topology from Istio metrics. Since service mesh acts as a transparent observability layer, Istio can automatically provide service topology information with minimal to no modification of the services themselves. All you have to do is enable Istio metrics and add a visualization layer on top of Istio.

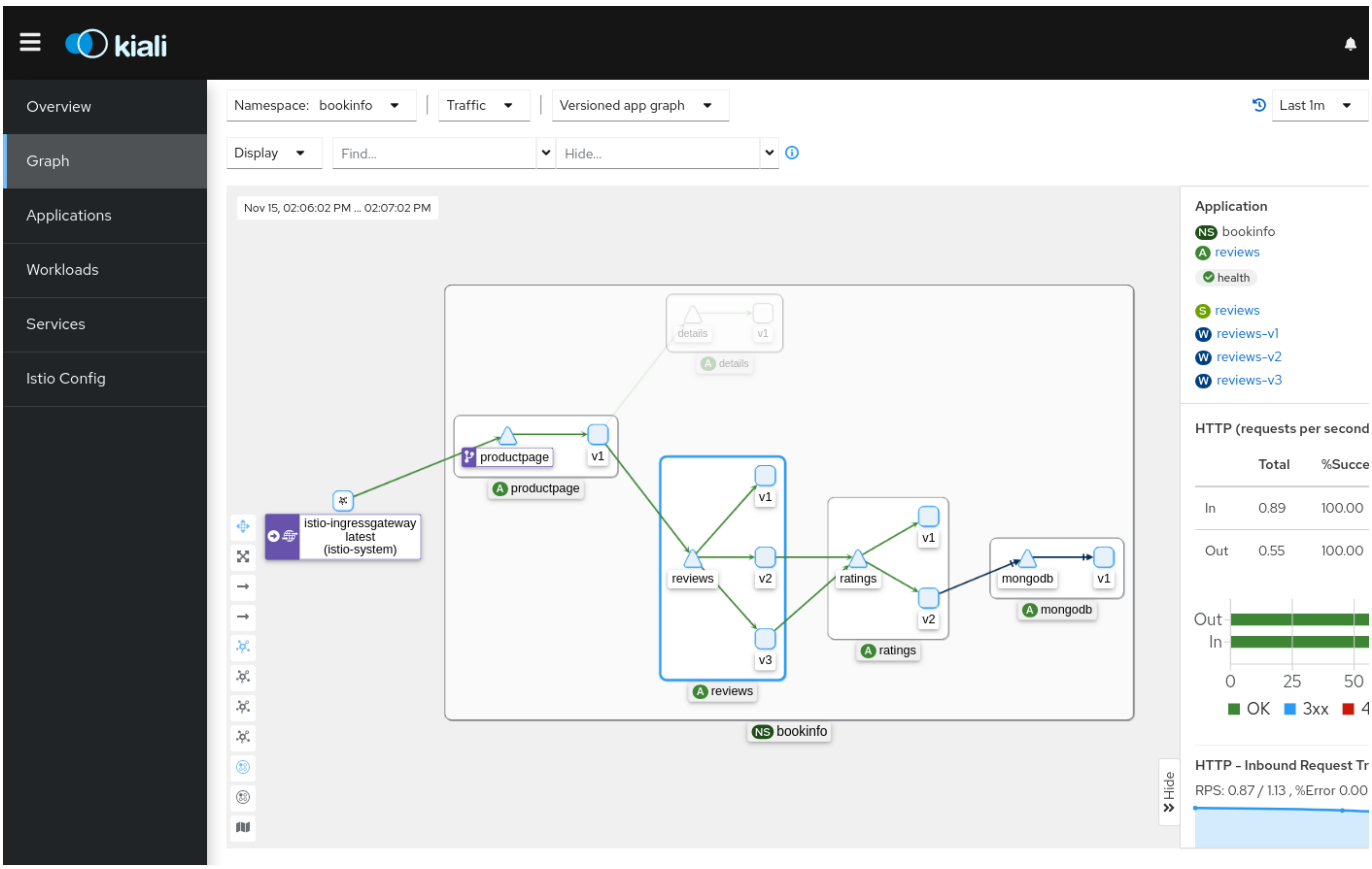


Example service graph

Source: Istio

How to Get Started

Even though Istio does not come with a user interface to visualize or drill down service topology, the open source tool Kiali is commonly integrated with Istio for service graph visualization. Kiali is available as an Istio add-on with community support, it is maturing quickly with fast release cycles. Similar to Grafana and Prometheus, many teams also build a management plane that handles the integration and lifecycle of Kiali once they reach scale. See the end of this whitepaper for support and out-of-the-box management plane options.



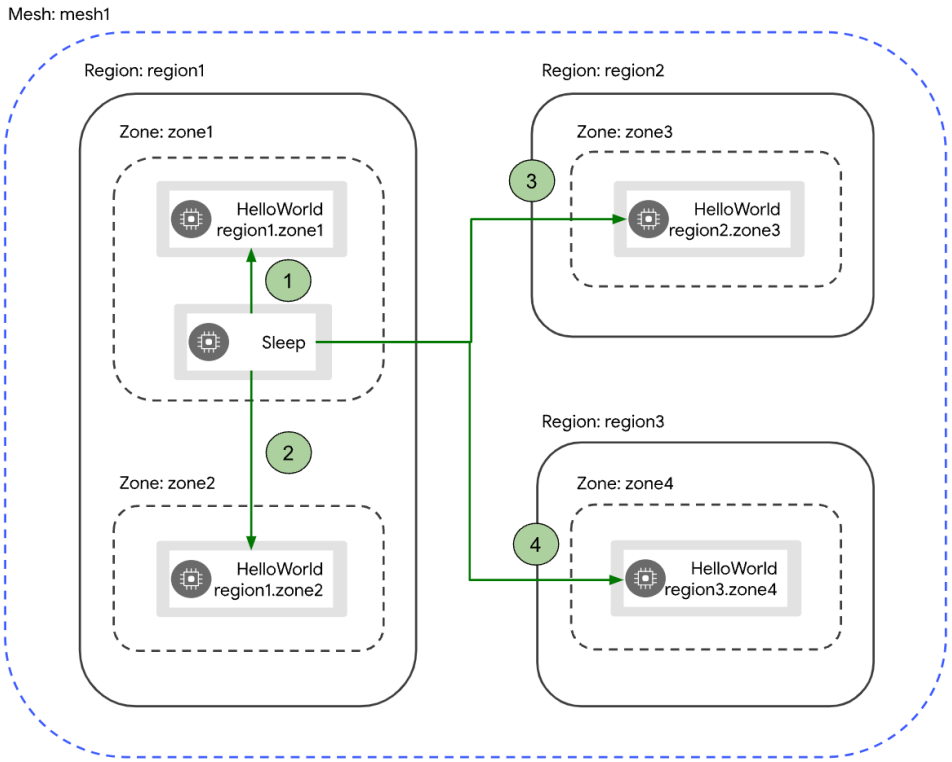
Service graph visualization in Kiali

Source: Istio

Strategy 3:
Set up Automatic Failover

Service Mesh Benefits

Outages can help at the service, zone, or even regional level. For example, even though AWS regions have unplanned outages very infrequently, the cost of an outage can be extremely high. Aside from AWS outages, there are many reasons outages can occur. A service mesh allows you to monitor issues as they emerge and set policies that automatically recover from failure by directing traffic to available services. With Istio, you can set up monitoring of failure rates across your services, and trigger a failover to the next locality as needed.



Service failover

Source: Istio

How to Get Started

Setting up failover in Istio requires proper configuration of Outlier Detection, Failover policies, and Connection Pool policies in Isito and Envoy. To properly test the failover, you can use the fault injection capabilities of Istio to generate failure scenarios. You can also use native AWS services such as Route53 to facilitate failover between AWS regions. Often teams will build a management plane to handle the integration of native AWS services with Istio to simplify failover configuration. See the end of this whitepaper for support and out-of-the-box management plane options.

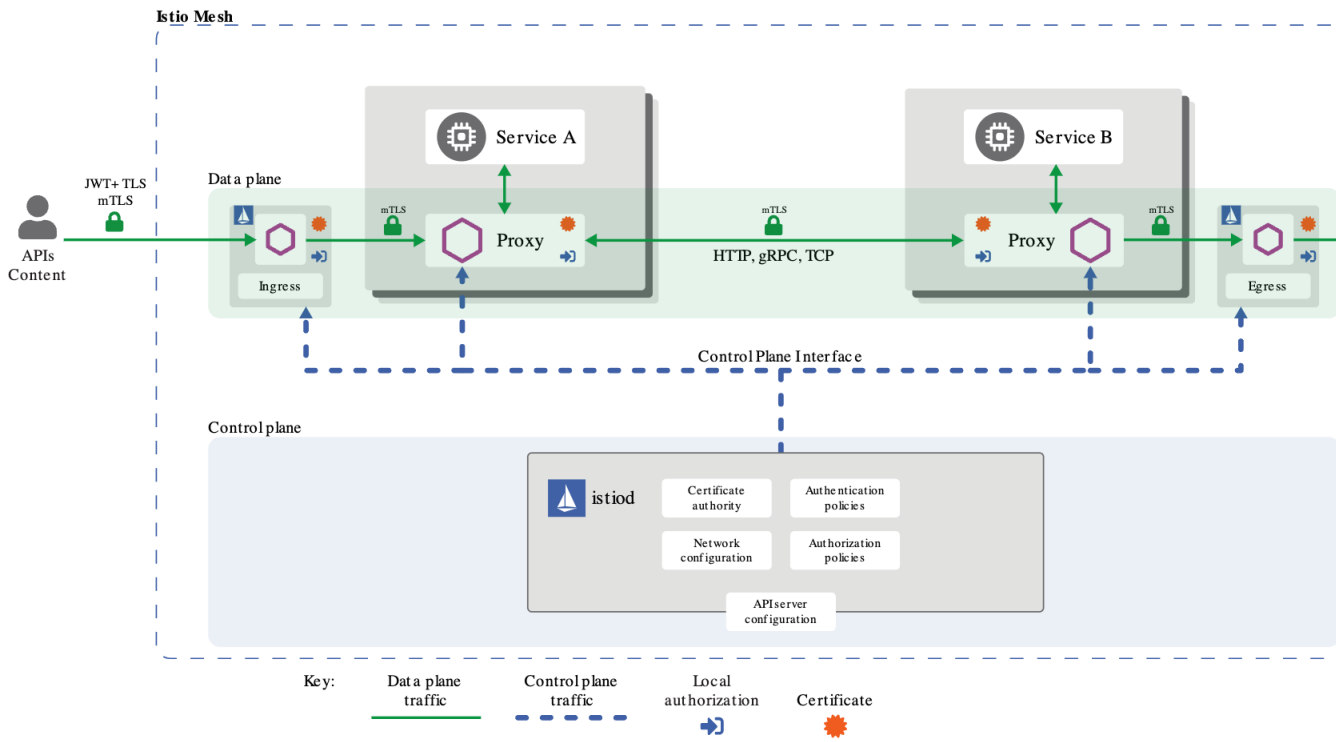
Strategy 4:
Set up a "Deny-All" Default Traffic Policy between Services

Service Mesh Benefits

Zero trust principles require a system to never assume trust, continuously verify trust, enforce least privilege, and remediate threats. "Never assume trust" in practice means implementing a "deny-all" policy by default in your EKS clusters. This means your services should not be able to connect unless they meet the conditions in which the requests are allowed — as defined by you.

Traditionally, implementing a deny-all policy translates to a very heavy configuration lift for the networking team. A service mesh simplifies this because Istio comes with key components:

- A Certificate Authority (CA) for key and certificate management
- The configuration API server distributes to the proxies, which work as Policy Enforcement Points (PEPs) to secure communication between clients and servers.
- A set of Envoy proxy extensions to manage telemetry and auditing



Istio security architecture overview

Source: Istio

How to Get Started

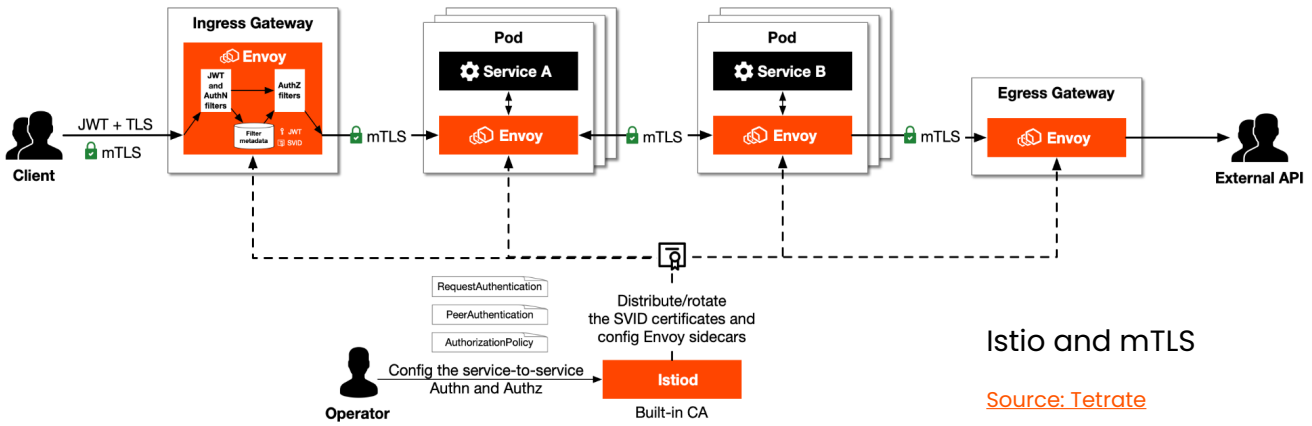
By default the Istio CA generates a self-signed root certificate and key and uses them to sign the workload certificates. To protect the root CA key, you should use a root CA which runs on a secure machine offline, and use the root CA to issue intermediate certificates to the Istio CAs that run in each cluster. An Istio CA can sign workload certificates using the administrator-specified certificate and key, and distribute an administrator-specified root certificate to the workloads as the root of trust. For a production cluster setup, you should use a production-ready CA such as AWS Private CA. AWS Private CA is for organizations building a public key infrastructure (PKI) inside the AWS cloud and is intended for private use within an organization.

After you've set up your CA, you can define Istio's peer and request authentication policies with YAML files, which are saved in the Istio configuration storage once deployed. Similarly, you can also define Istio authorization policies using YAML files. The authorization policy enforces access control to the inbound traffic in the server side Envoy proxy. Each Envoy proxy runs an authorization engine that authorizes requests at runtime. When a request comes to the proxy, the authorization engine evaluates the request context against the current authorization policies, and returns the authorization result, either ALLOW or DENY. As policies proliferate, many organizations build a management plane on top of Istio to simplify lifecycle management of policies. See the end of this whitepaper for support and out-of-the-box management plane options.

Strategy 5:
Implement mTLS across Services without Slowing Down Application Development

Service Mesh Benefits

Many organizations want to implement Zero Trust measures such as mTLS, but are deterred by the cost to appdev velocity — meaning they cannot afford to prioritize the implementation of "policy code" for each service, as the cost of doing so on each app team is too high. Service mesh simplifies rolling out mTLS between services by acting as a network overlay. Specifically, Istio and Envoy perform all the heavy lifting including: traffic forwarding from app to Envoy sidecar, provisioning strong identities to every workload with X.509 certificates, and managing the mutual TLS connection. Istio can be configured to work with a wide range of external CAs, including AWS Private CA.



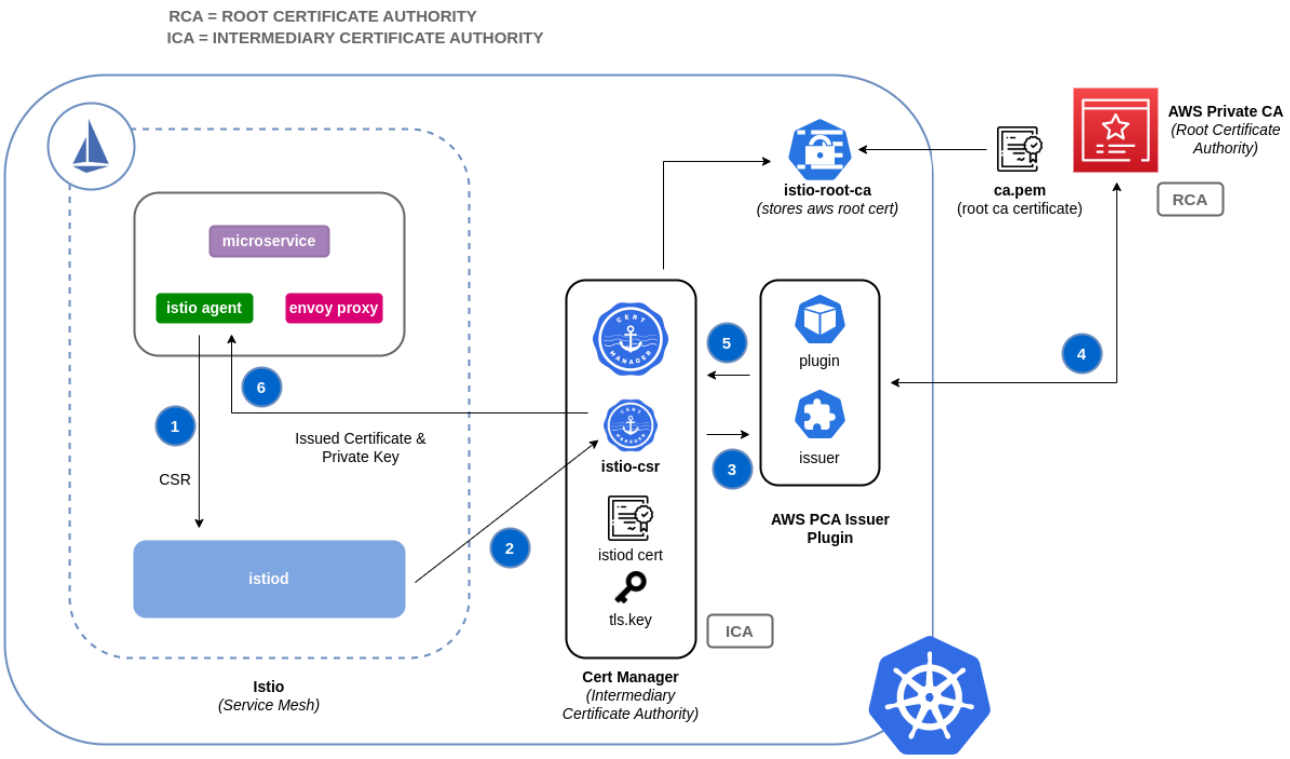
Istio and mTLS

Source: Tetrate

How to Get Started

You can enable TLS and mTLS to secure your EKS application workloads at the ingress, on the pod, and between pods using AWS Private CA. AWS Private CA is for organizations building a public key infrastructure (PKI) inside the AWS cloud and is intended for private use within an organization. You will also need to install Kubernetes add-on cert-manager to distribute, renew, and revoke certificates. You can also use AWS Private CA with AWS Certificate Manager (ACM) to manage certificate issuance and automate certificate renewals. To use AWS Private CA with Istio and cert-manager in your EKS cluster, you can disable the Istio control plane (specifically istiod) from functioning as the root Certificate Authority (CA), and configure ACM Private CA as the root CA for mTLS between workloads.

Given the complexity of integrating various open source and AWS native services, many organizations maintain a dedicated cloud platform team to support this at scale. Alternatively, see the end of this whitepaper for support and out-of-the-box management plane options.



Istio with AWS Private CA

Source: AWS

Getting Started with Istio and Envoy on EKS

How to get started depends on where you are in your journey and your scale.

For Teams Experimenting in a Single EKS Cluster
The easiest way to install and try Istio on EKS is through a native EKS add-on—Tetrate Istio Distro (TID). Currently it is the only 100% upstream Istio distribution available as an EKS add-on. Installing TID on your EKS cluster is as simple as one command. Tetrate also provides Tetrate Istio Subscription (TIS) when you run into issues. TIS includes a global team of experts to answer your service mesh questions, architecture review, and extended support plus CVE fixes to unblock you during your service mesh design.

For Teams Seeking Enterprise Support
For organizations that require enterprise support for the production deployment of open source software, Tetrate Istio Subscription (TIS) provides extended support and CVE fixes so you can efficiently manage your open source upgrade cycles. TIS also includes access to a global team of experts to answer your service mesh questions. Tetrate's premium enterprise support offers 24 x 7 support for severity 1 and 2 cases, the next business day for severity 3 cases, and two business days for severity 4 cases. For sev 1 cases, Tetrate's premium support team responds within one hour to ensure your production environment stays operational.

For Teams Taking Istio into FedRAMP Environments
If you are an advanced Istio user planning to take your deployment into FedRAMP, you will need a FIPS-verified Istio to accelerate your roadmap. Tetrate Istio Subscription (TIS) offers a FIPS-verified distribution of Istio that has extended support and CVE fixes. Contact Tetrate to learn more about the FIPS Istio distribution in Tetrate Istio Subscription.

For Teams Scaling across Multiple EKS Regions
When you need to take Istio beyond one EKS cluster into multiple regions, you need a management plane to coordinate multiple Istio control planes to achieve end-to-end security and failover. Tetrate Service Express (TSE) provides a management plane that natively integrates with AWS services such as PCA, Route53, and NLBs to give you security and availability, as well as the observability to optimize costs out of the box. Tetrate provides a 30-day evaluation of TSE, sign-up on Tetrate's website for free access.