tetrate

The Essential Service Mesh Adoption Checklist

A best practices guide to help security, networking and platform teams successfully deploy the service mesh.

2023 Edition

Table of Contents

Introduction

By offering connectivity, reliability, observability and security, the service mesh has evolved into a fundamental component of an organization's application and infrastructure modernization initiatives. In general, your organization can reap substantial benefits from implementing a service mesh, especially when dealing with distributed applications composed of many microservices. As application traffic grows, requests between these services can increase exponentially, requiring sophisticated routing capabilities to optimize the flow of data between the services and ensure the application continues to perform at a high level. From a secure communications standpoint, service meshes are essential for enabling the secure TLS (mTLS) connections between services.

Because service meshes manage the communication layer, they liberate developers from the complexities of managing how each service interacts with all the others. This allows developers to dedicate more time to adding business value with each service they create. For DevOps teams that have an established production CI/CD pipeline, a service mesh can be essential for programmatically deploying apps and application infrastructure (Kubernetes) to manage source code and test automation tools. Additionally, a service mesh enables DevOps teams to manage their networking and security policies through configuration, enhancing overall operational efficiency and ensuring consistent policy enforcement.

Depending on the structure of your organization, one or more teams or roles may be responsible for installing and configuring a mesh. Based on our experience with customers, we've developed best practice guidelines for deploying the service mesh successfully. (Tetrate's service mesh is based on the open source Istio service mesh.) This checklist, outlining the essential steps for a successful service mesh adoption journey, will ease the transition as you modernize applications with a microservices architecture and help you achieve ROI today. Addressing each one will not only smooth your adoption of Istio but will help transform your organizational culture and simplify workflows across the enterprise.

Before you begin your service mesh journey, remember three things:

1 Security, networking, operations and platform teams face new challenges when managing cloud-native, microservices environments.

Today, leaders across the organization face numerous challenges brought about by the increasing complexity of distributed microservices environments. From ensuring system stability to driving innovation, they must navigate a landscape that demands seamless performance, scalability, zero trust security and reliability. The intricate nature of these environments poses significant challenges in terms of visibility, troubleshooting, policy management and efficient resource utilization. Addressing these challenges requires a combination of effective tools, practices and collaboration among security, networking and platform teams. Additionally, a strong understanding of microservices architecture and its associated complexities is crucial for success.

2 The service mesh is vital to streamline the process of managing microservices and improve DevOps efficiency.

To address these challenges, you can use a service mesh: a dedicated infrastructure layer that enables teams to manage and secure network connectivity between services across multi-cloud environments and runtimes. A service mesh transparently oversees and monitors all traffic for your application, typically through a set of network proxies that sit alongside each microservice. Adopting a service mesh allows you to decouple your application from the network, and in turn, allows your operations and development teams to work independently.

3 Invest in training and education.

To successfully integrate service mesh into your platform engineering strategy, your team must have the necessary skills and expertise. Invest in training and education programs to ensure your team is well-equipped to manage and maintain the service mesh infrastructure.

Preparing Your Teams

The rapid adoption of cloud-native technologies has transformed the way organizations build, deploy and manage applications. The service mesh is an essential component of modern, cloud-native architectures, providing enhanced observability, fine-grained traffic control and robust security features. Not only does the service mesh provide vital features, it does so in a way that's global, uniform and independent of the application.

To get started with the service mesh, it is essential to align your security, networking and platform teams. Begin with a small number of services or a less critical environment to gain experience and confidence in Istio's capabilities. This approach allows you to identify and address any issues early in the process and to understand how the service mesh will fit into your existing infrastructure and processes. You'll know you're ready to move forward when you've established a plan for each of the key areas we list below and the teams you're working with are committed - and, ideally, excited - to adopt the mesh and experience its benefits for themselves.

Security

Work with the security groups and the CISO office to:

Build the mesh security model for your organization. A few important areas to explore:

- [ ] How does the mesh fit in with the DMZ?
- [ ] How does the mesh fit with the existing firewall/WAF?
- [ ] How does the mesh fit with existing SSL termination?
- [ ] How does the mesh fit with existing API Gateway appliances?

Develop a PKI/certificate strategy for the mesh. Determine what integration work is needed:

- [ ] Will the mesh issue certificates out of the organization's existing root, or from a separate root of trust?
- [ ] Will the mesh issue certificates or will you use existing certificate infrastructure for workload certificates? Depending on the approach, this may require changes to those certificates for the service mesh's authentication capabilities to work.
- [ ] How will we choose to deliver signing certificates to the mesh control plane?

Determine what policies the mesh will enforce from the outset.

Onboard service mesh containers into your system, with all required security scanning.

- [ ] Put a process in place to keep these images up-to-date at a regular cadence. For reference, Istio releases four times a year.
- [ ] Put a process in place to address CVE/image scanning and pentest results for the mesh infrastructure.

Networking

Work with the networking team to:

Assess how the mesh changes networking models in your organization:

- [ ] For example, you may eliminate hairpin traffic from apps that communicate "internally" by going out to a global load balancing system and back through the DMZ and into the datacenter, even when the two workloads run on the same rack. This will have an impact on networking and security.

- [ ] Determine where TSL (SSL) termination needs to happen. In existing load balancers, offload it to the mesh, or even the application.

- [ ] Determine how you will manage and deploy the mesh's gateways.

- [ ] Determine your gateway deployment model: one gateway per team; a single shared gateway for all teams; or mostly shared, with a few standalone. We recommend a gateway per team to start with.

Develop a strategy for how to integrate the mesh with other networking components.
For example, how does the mesh fit with existing API Gateway appliances?

Capacity Planning

Work with your capacity planning teams to ensure you're ready for the service mesh.

- [ ] The overhead of the service mesh can vary, but we typically see ~10% overhead, with some applications seeing as much as 30% depending on use cases and workloads. In our experience, existing systems tend to be overprovisioned to the extent that the mesh infrastructure can be added without meaningful changes to resource planning in the short to medium term.

Operations

Work with the operations team to prepare them for incoming infrastructure changes.

- [ ] Start to familiarize them with the service mesh and the capabilities it brings if they were not part of the buying decision.
- [ ] Begin education and training with them, the earlier the better.

Observability and Monitoring

Work with your observability and monitoring teams to prepare them for incoming infrastructure changes.

- [ ] Determine how you'll integrate the mesh's metrics with the existing infrastructure.
- [ ] Ensure you have capacity for the additional metrics the mesh will produce.
- [ ] Evaluate log levels and the resulting log volume produced by the mesh. Verbose logging in the mesh can produce a lot of data!

Platform

Prepare your larger platform team for the incoming infrastructure changes.

- [ ] Get every team member hands-on with your chosen mesh solution.
- [ ] Begin education and training for the platform team ASAP.
- [ ] As a goal, we recommend that a handful of platform team members engage in the underlying OSS projects. This will increase organizational familiarity, confidence, and ability to operate with those technologies.
- [ ] Develop the tenancy model for your system:
  - [ ] Do teams own namespaces? Whole clusters?
  - [ ] How does that map to the mesh?
  - [ ] Will you share ingress gateways or give an ingress gateway per team? We recommend a dedicated ingress gateway per team to start with.

Initial Onboarding

Once your teams are prepared and ready to start implementation, it's time to start the initial phase of adoption.

Bucket your applications by criticality, runtime requirements, frameworks and protocols.

- [ ] Determine the order in which you'll attack the buckets. We recommend starting with simpler applications that have low criticality and more relaxed runtime requirements—e.g., non-PCI or with loose latency requirements.
    - In the organizations we work with, this group often includes the first Kubernetes services being developed.
    - If your first bucket contains a large group of applications, we recommend hand-selecting an initial subset—between three to five teams—to work with closely and directly.

Begin to roll out the mesh one bucket at a time:

- [ ] Work closely with your initial teams to develop tools, practices, process, and expertise for this class of applications.
    - [ ] Ingress gateway strategy: shared or dedicated to the team?
        - Decide based on criticality and production experience; favor dedicated gateways until you're very confident.
        - Highly critical applications should almost always get a dedicated gateway.
    - [ ] CI/CD changes.
        - If the application team will be authoring mesh configuration, that configuration should be deployed by the same continuous deployment system that rolls out their application.
    - [ ] Understand any application changes that may be required:
        - See Istio's documented pod requirements.
        - Do you need to change the application's port labels, or turn off TLS in the application to allow the mesh to do it?
        - How do we deploy the sidecar itself? Do they opt in? Opt out?
        - For this set of apps, are there any systemic changes that need to be made—e.g., exempt database traffic for the time being?
- [ ] Deploy dashboards for the application based on mesh metrics.

Execute the mesh onboarding. This will usually take a few deployment cycles, as you want to gather information and expertise before, for example, requiring mTLS everywhere. The major steps for each application are:

- [ ] Start with PERMISSIVE mTLS for the application.
    - This will ensure non-mesh applications calling this application do not break.
- [ ] Configure gateway routing.
    - If you decide to use dedicated gateways, deploy the gateway instances themselves.
- [ ] Add sidecar proxies to the application instances (enable injection).
- [ ] Add a DestinationRule requiring mesh mTLS.
    - This ensures applications in the mesh are using mTLS even if the server allows non-mTLS clients as well.
- [ ] Configure traffic management settings.
- [ ] Configure rate limiting and authentication policies.
- [ ] Author authorization policies restricting which clients can communicate.
    - These are typically written by the app team.
- [ ] Lock down STRICT mTLS as soon as all clients have upgraded.

Then apply those learnings to the entire bucket of apps.

- [ ] Create a rollout template that encodes the happy path for all applications in that bucket.
- [ ] Create playbooks for executing the onboarding.

Repeat with the next bucket of more critical, higher runtime requirement, “harder” applications, using the previous learnings to make it easier and faster to execute.

About Tetrate

Rooted in open source, Tetrate was founded to solve the application networking and security challenges created by modern computing so enterprises can innovate with speed and safety in hybrid and multi-cloud environments. As applications evolve into collections of decentralized microservices, monitoring and managing the network communications and security among those myriad services becomes challenging. This is why some of the largest financial institutions, governments and other enterprises rely on Tetrate to deliver modern application networking and security on a foundation of Zero Trust.

Find out more at www.tetrate.io.

Tetrate Academy

Accelerate your service mesh journey with expertly curated, hands-on training courses from the co-creators of open source Istio and Envoy. Private training for enterprise customers available upon request.

Learn more at academy.tetrate.io