WHAT IS A SERVICE MESH?

Organizations are using microservice architecture to build scalable network applications thanks to the resurrection of containers. In a typical microservice architecture, the application consists of multiple microservices that communicate using remote procedure calls (RPC) over the network, with each service performing a specific business task. This allows each of the services to be built using whatever programming language fits the task, a backend datastore that fits the data type, and independent deployment (each service has an independent lifecycle).

One of the biggest benefits you get when using this pattern is that you can have different developers working on different services simultaneously. In the case where one service goes down, the impact is limited to that service, not the others.

Of course, microservices as a pattern does come with some disadvantages, especially when you run a large ecosystem of services. The bigger the ecosystem, the more difficult it is to get insight into all the moving pieces and to manage the network communications between all the services.

This is where service mesh comes in.

What is a service mesh?

A service mesh is a dedicated infrastructure that handles all service-to-service network communication needs within the ecosystem. It provides visibility, resiliency, traffic, and security control of these services with little or no change to your existing code. This is a big win for developers, who no longer have to build networking concerns into their code.

Advantages of using a service mesh

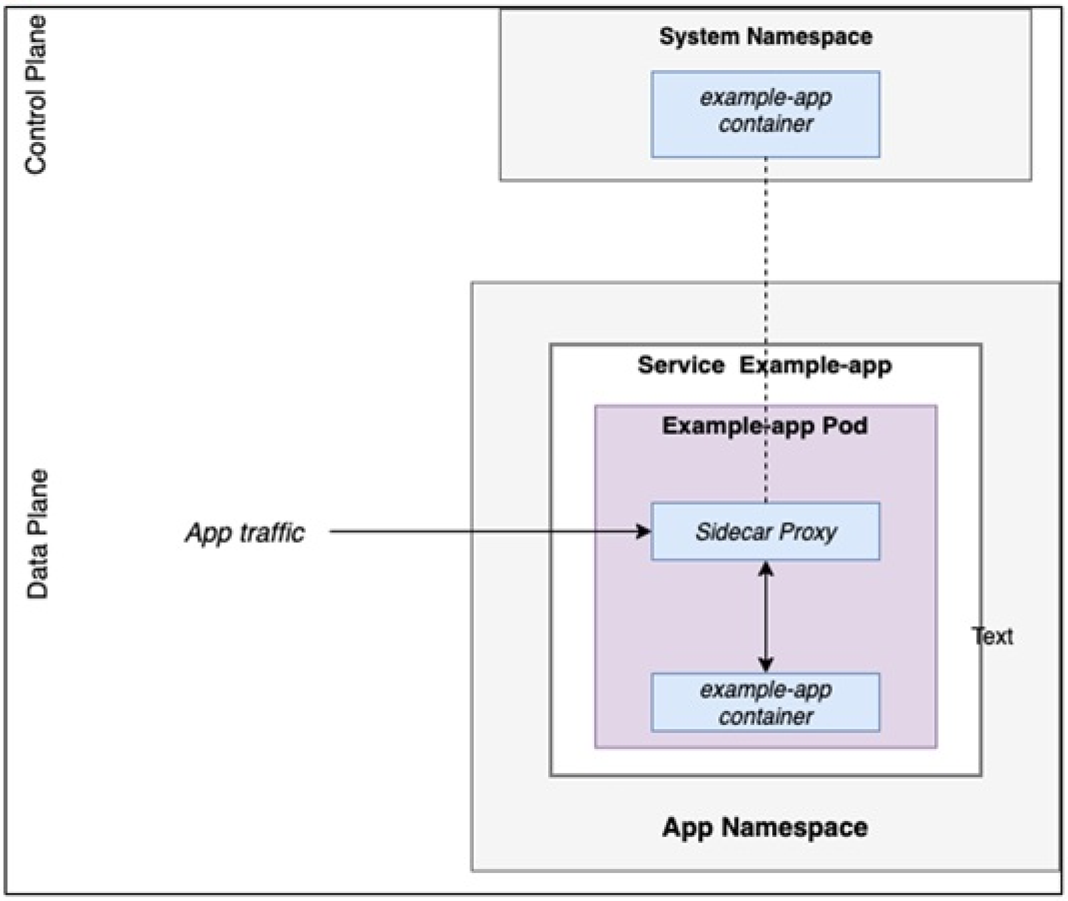
Using a service mesh offers plenty of benefits, including:

- **Observability.** Typically, teams have different tools for keeping track of logging, tracing, metrics, and things like security control. With a service mesh, you get all these benefits, eliminating the need for other tools.
- **Resiliency.** Service mesh offers resiliency features like circuit breaking, latency-aware load balancing, eventually consistent service discovery, retries, timeout, and deadlines.
- **Traffic control.** A service mesh gives granular control of network traffic to determine where your request is routed.
- **Security.** Service mesh provides certificate authority that generates a per service certificate for TLS communication between services.
- **Delay and fault injection.** Service mesh allows you to inject failure and latency to simulate what would happen in a real situation. The benefit is that you can rest assured that you know how your services will behave in case of an issue.
- **Less code for Devs and Ops.** Service mesh also reduces the amount of code devs must write. Ops don't have to involve the devs about things like service retries and timeouts. A dedicated layer of infrastructure that takes care of this sits between the devs and ops, allowing a uniform and yet highly customizable way of handling service-to-service communication while allowing the code to stay relatively light.

Service mesh architecture

A typical service mesh architecture is made up of two network planes.

- The **Data Plane** (proxy) is responsible for moving application request traffic between all the services. This plane is responsible for things like health checking, load balancing, authentication, authorization, etc.
- The **Control Plane** provides policy and configuration for all the services in the mesh. It does not touch or care about the network packet, its main job is to configure the data plane.



Example: Service mesh architecture in a Kubernetes cluster

Popular service mesh projects

Several technologies offer service mesh technologies. The most popular ones are:

- Istio
- Linkerd
- Envoy
- Conduit (now a part of the Linkerd project as Linkerd 2.0)