SUSEcon digital '20

Managing Microservice Applications with Istio

Connect, Secure, Collect metrics and Monitor applications in a complex environment

DEV-1393

Swaminathan Vasudevan

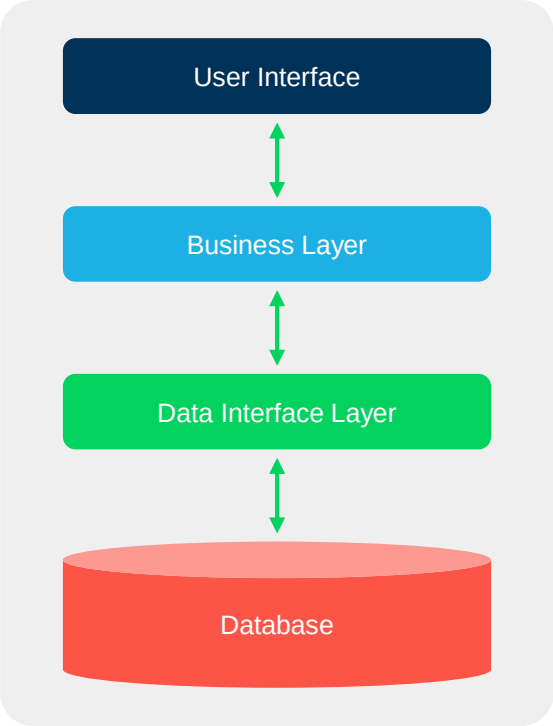
SUSE

Need For Microservices

What Is A Monolithic Application?

- Monolithic applications are built as one single unit.
- Enterprise monolithic applications mostly have a three-tier architecture (user interface, server, and database).
- A developer must build and deploy an updated version of the monolithic application for any changes that needs to go into any one of the layers.

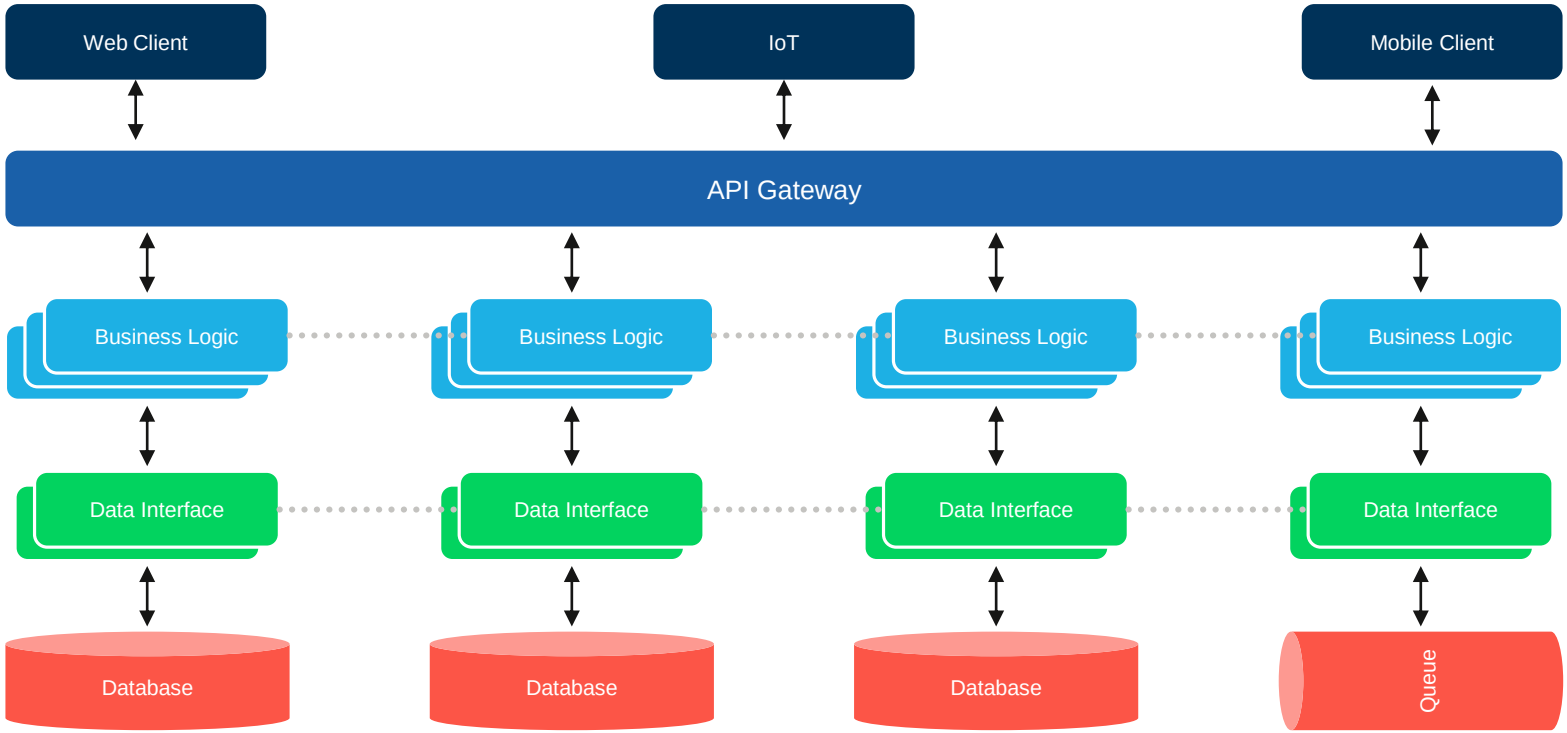
Monolithic Application



Disadvantages Of A Monolithic Application?

- Scaling of monolithic applications is always a challenge.
- As monolithic applications grow larger in size and complexity, the complete picture becomes difficult to understand by the developers.
- Limited reuse is seen across monolithic applications.
- Difficult to achieve operational agility in repeated deployments.
- Monolithic applications are developed using single stack, this limits the use of possible other available tools.

Microservices Architecture



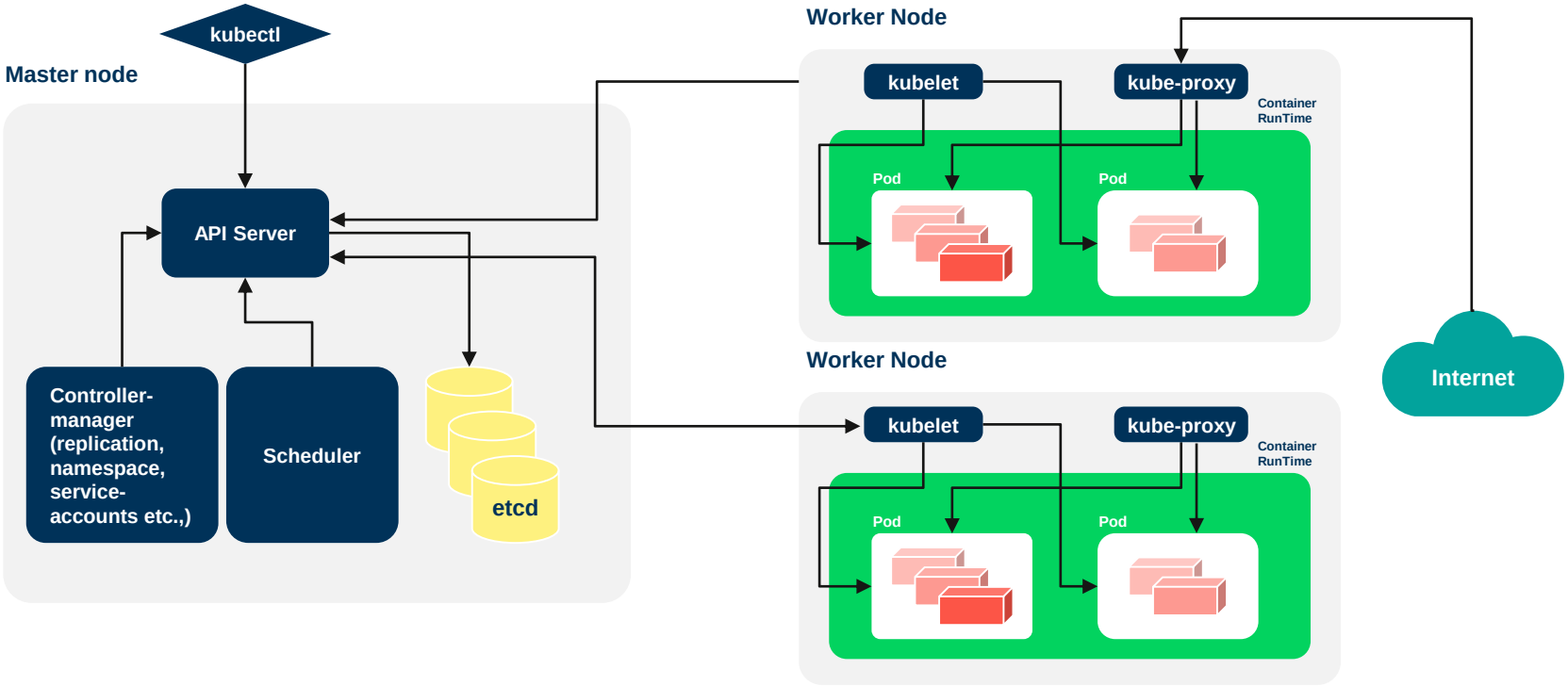
Benefits Of Microservices Architecture

- Enables an application to be broken down into multiple component services, allowing each of these services to be deployed and maintained without compromising the integrity of the application.
- Enables continuous delivery.
- Provides scalability and reusability with efficiency.
- Each service can be developed and deployed independently.
- Better fault isolation.

Benefits Of Microservices Architecture

- Simplifies security monitoring because the various parts of an app are isolated.
- Components can be distributed across multiple servers or even multiple data centers.
- Complements cloud activities and works with containers.
- Code can be organized around business capabilities.
- With microservices, interfaces are exposed with a standard protocol, such as a REST-ful API, so they can be consumed and reused by other services and applications without direct coupling through language bindings or shared libraries.

Kubernetes Architecture And Control Flow



Challenges With Microservices At Scale

Things To Consider For A Complex, Scalable, But Resilient Microservice Architecture

Kubernetes provides a framework to deploy and manage container applications that are part of a microservice architecture.

But as the number of service increases, we need to deal with:

- Interaction between the services
- Security between the services
- System health
- Fault tolerance

Things To Consider For A Complex, Scalable, But Resilient Microservice Architecture

- Logging
- Telemetry
- Metrics
- Circuit breaking
- Multi-point failures and more

Service Mesh

Service Mesh Comes To The Rescue

What is a Service Mesh?

A Service Mesh addresses the challenges developers and operators face as monolithic applications transition towards a complex distributed microservice architecture.

A Service Mesh decouples this complexity from your application and puts it in a service proxy -- it handles it for you.

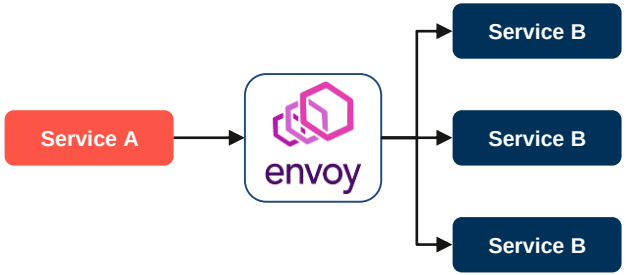
Service Mesh Comes To The Rescue

Service proxies offer functionalities like:

- Traffic management
- Circuit breaking
- Service discovery
- Authentication
- Monitoring
- Security and much more

What Is Envoy?

- Service and Edge Proxy
  - HTTP/2, gRPC, MongoDB, DynamicDB with more protocol support in future
- Advanced load-balancing
  - L7, Canary, Retries, Circuit breaking, Rate limits
- Security
  - Authorization, mTLS
- Observability
  - Tracing & metrics
- Extendable
  - Go extensions, WASM, LUA, etc.,



What Is Istio?

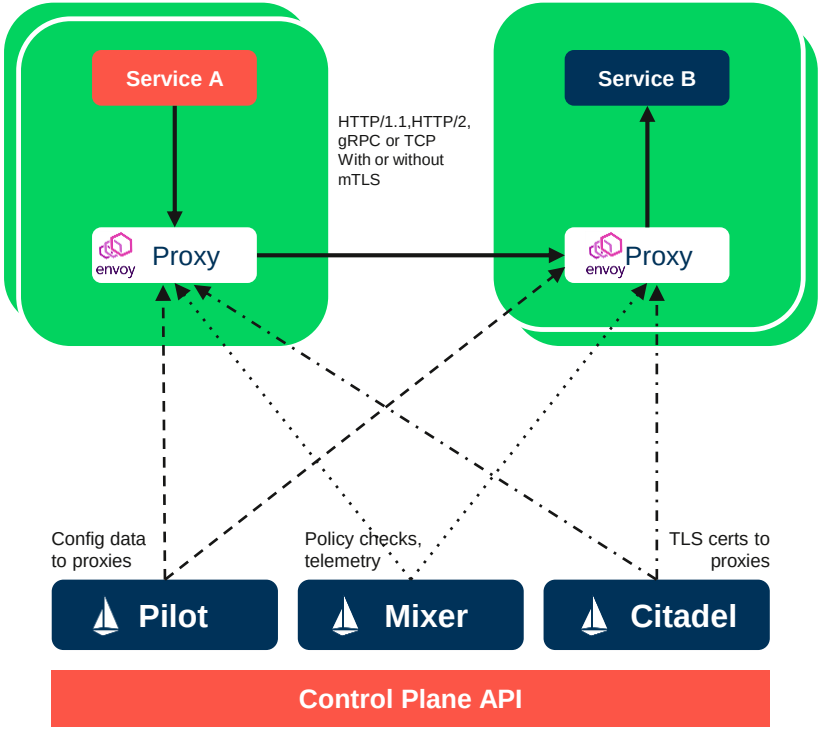
Istio is an open source service mesh that layers transparently onto existing distributed applications.

Istio provides an easy way to create a network of deployed services with

- Load balancing
- Service-to-service authentication
- Monitoring and more

without requiring any changes in service code.

Istio support can be added to services by deploying a special sidecar proxy between microservices, configured and managed using Istio's control plane.



Istio Control Plane

- Automatic load balancing for HTTP, gRPC, WebSocket, and TCP traffic.
- Fine-grained control of traffic behavior with rich routing rules, retries, failovers, and fault injection.
- A pluggable policy layer and configuration API supporting access controls, rate limits, and quotas.
- Automatic metrics, logs, and traces for all traffic within a cluster, including cluster ingress and egress.
- Secure service-to-service communication in a cluster with strong identity-based authentication and authorization.

Istio Control Plane Components

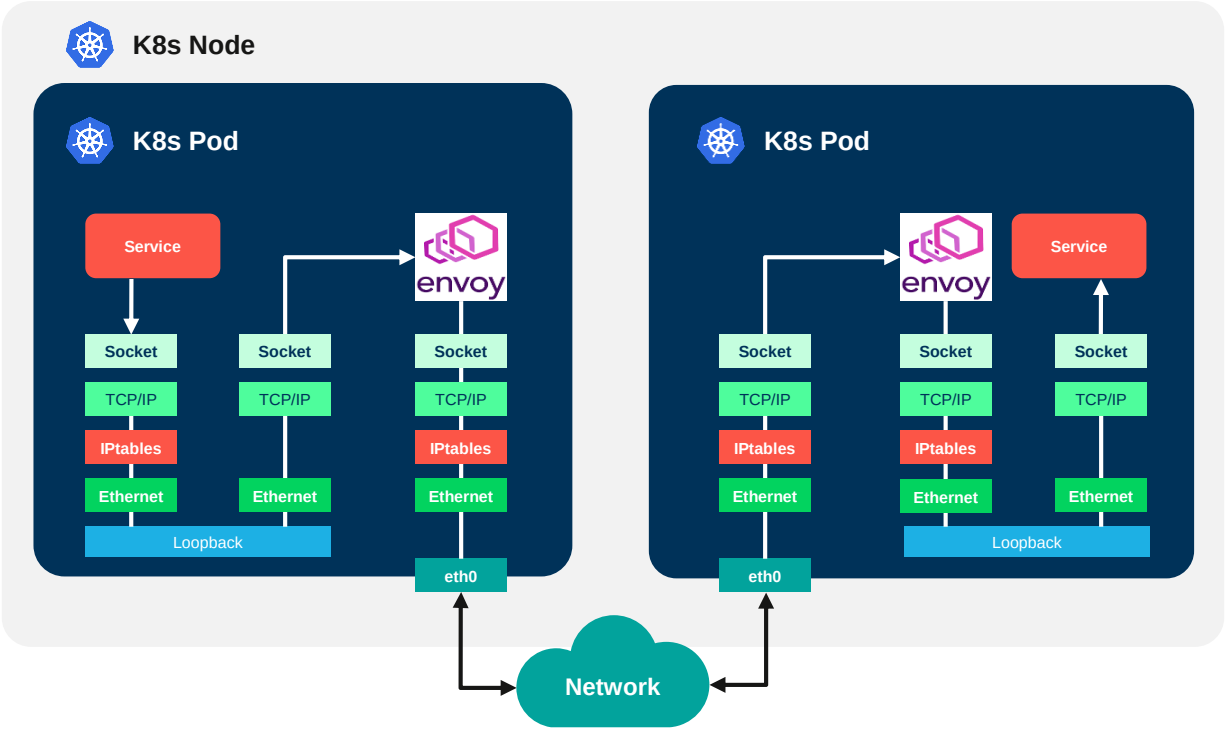
Istio control plane components are

- Pilot
  - Connectivity and Communication (Traffic Management, Fault injection and Layer 7 Load Balancing)
- Mixer
  - Monitoring and observability (backend abstraction, intermediation, latency, reliability)
- Citadel
  - Encryption and authentication (service authentication, role based access control, authentication policy, TLS authentication and key management)

Istio Data Plane

- Istio data plane consists of Envoy proxies that are deployed as sidecars within each container.
- These proxies are responsible for establishing connections between the services and managing the communication between them.

Transparent Sidecar Injection Without Cilium



Cilium With Istio

How Cilium Enhances Istio with Socket-aware eBPF Programs

Cilium and eBPF Programs enhances Istio:

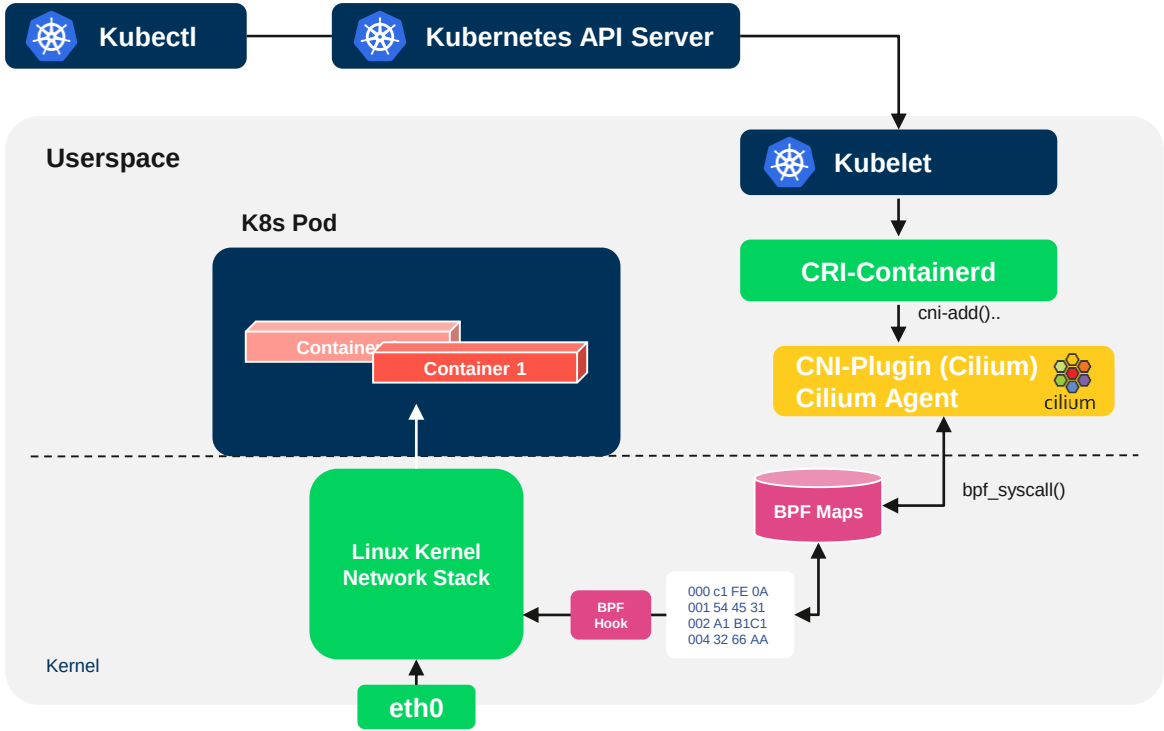
- Increase Istio security
  - Least privilege security for multi-container pods using socket-aware BPF programs
  - Protect from compromised sidecar proxies and protocols that bypass the sidecar
  - Use of BPF to force all application traffic through the sidecar proxy
- Enable Istio for external service
  - Using socket-aware BPF programs and kTLS to provide visibility and control into TLS encrypted connections
- Performance
  - Efficient networking and socket redirection to accelerate Istio

What Is Cilium?

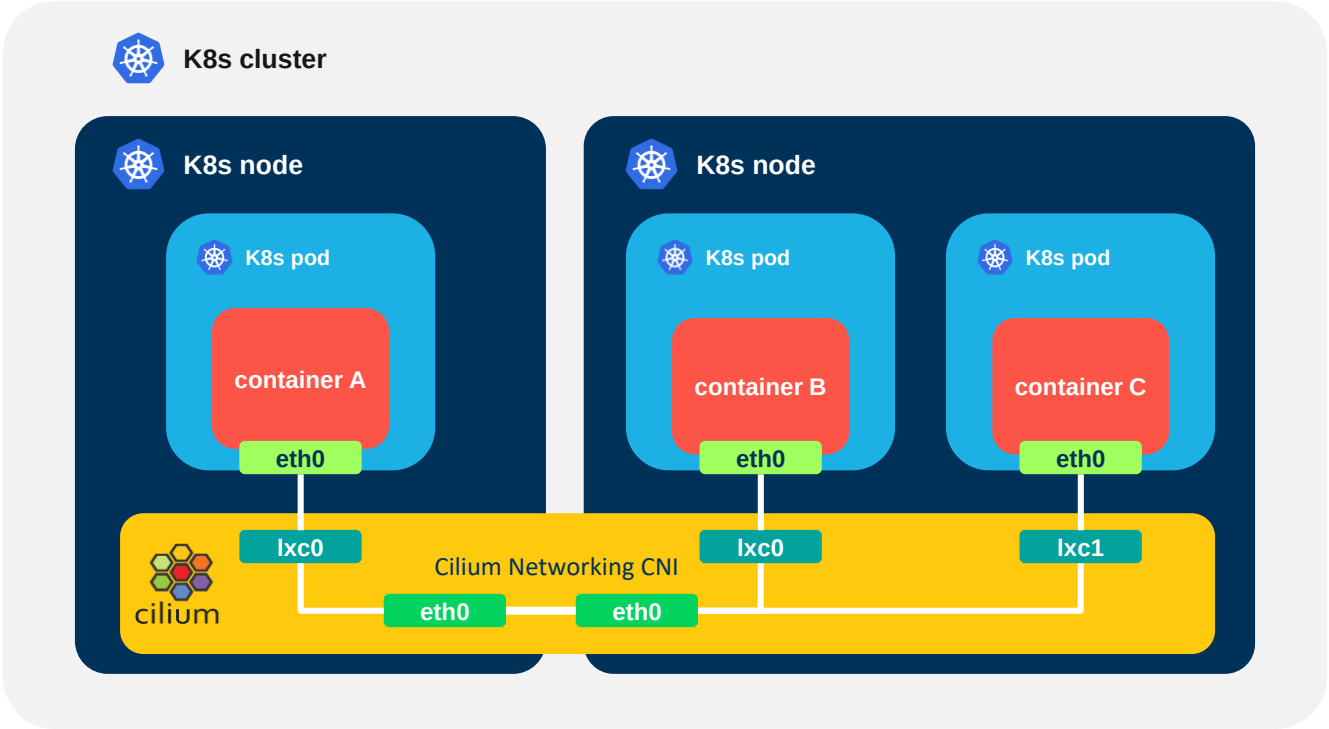
Cilium is open source software for transparently securing network connectivity between application services deployed using Linux container management platforms like Docker and Kubernetes.

- Highly efficient BPF datapath
  - Fully distributed
  - Service Mesh datapath
- Networking
  - Cilium-CNI or chaining on top of most other CNIs
- Kubernetes services implementation
- Network policies on both packet and API level
  - Identity-based, IP/CIDR as fallback, DNS aware, API aware
- Multi-cluster, encryption
- Available for Linux Kernel 4.9 and above.
- Distributed and scalable load balancing.
- Native Envoy and Istio integration
  - Transparent Envoy injection (per-node or sidecar)
  - Accelerated proxy redirection, transparent SSL visibility

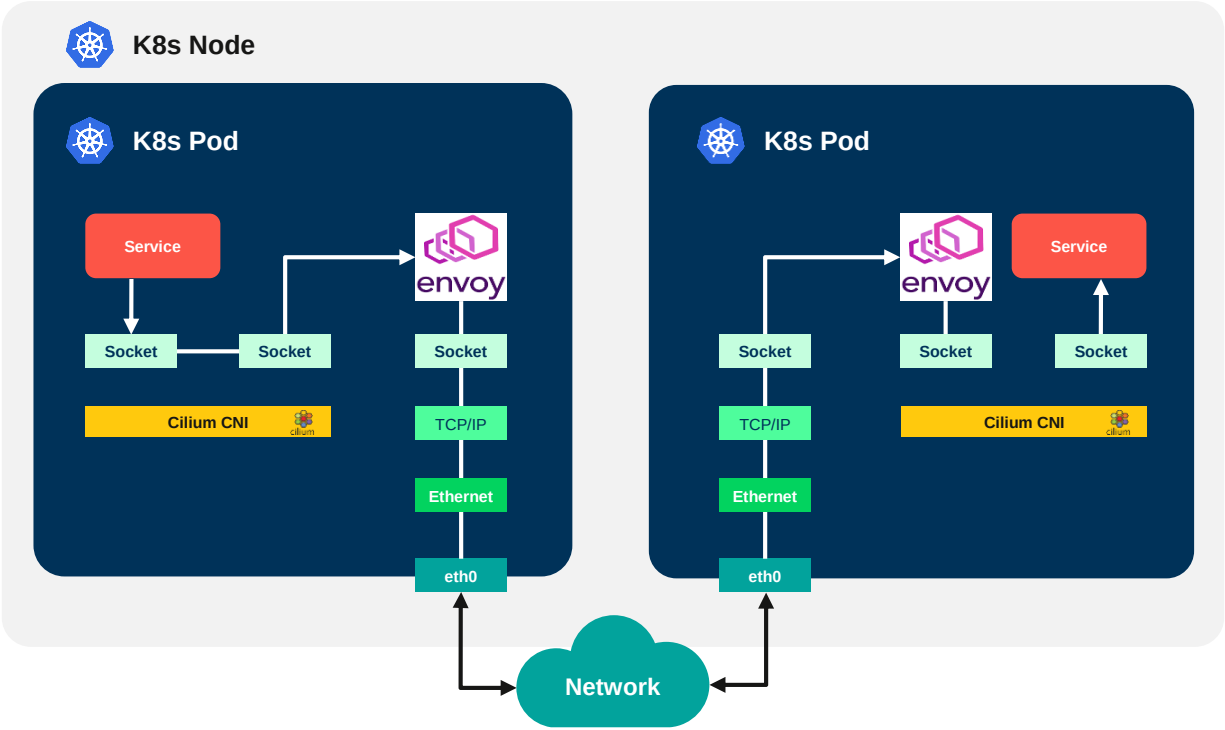
Kubernetes, Cilium CNI Control Flow



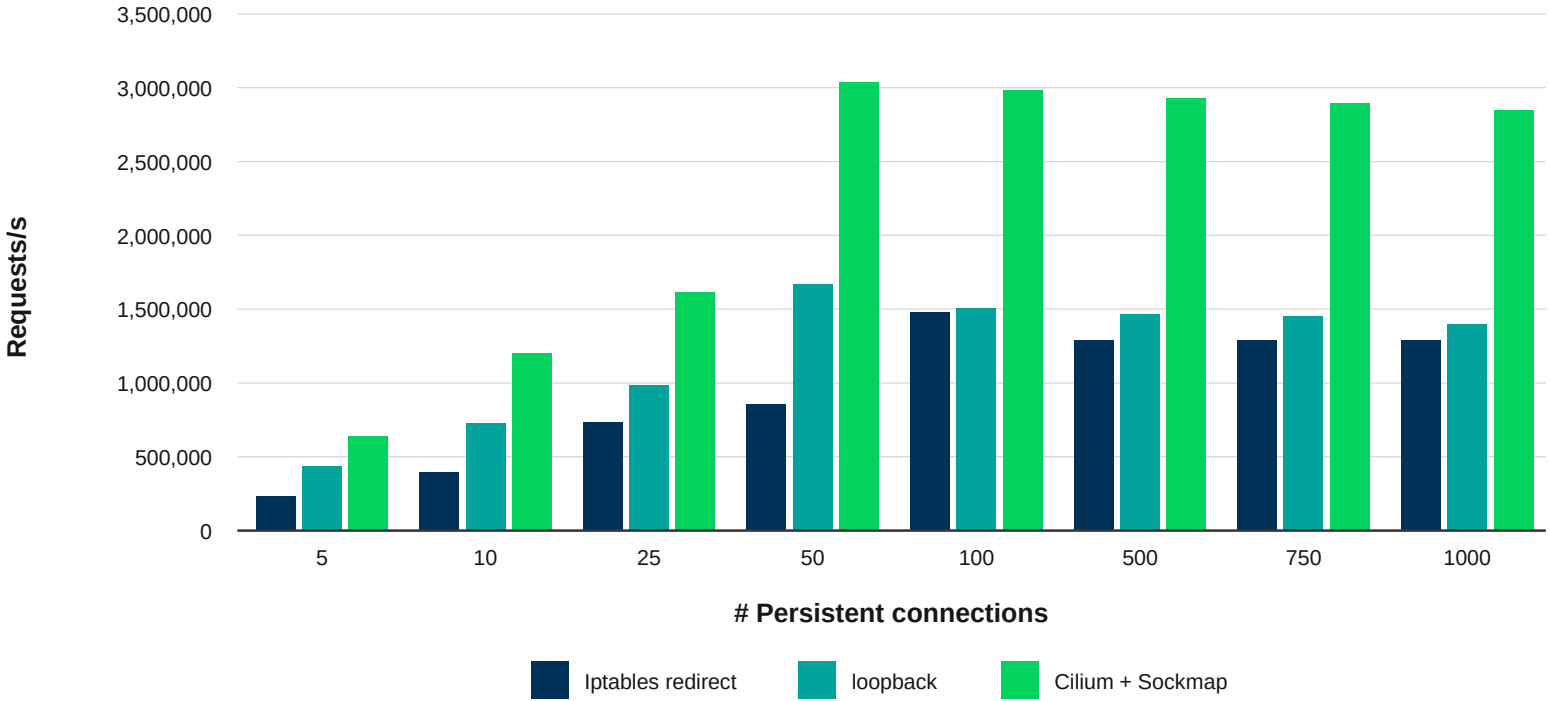
Kubernetes Cluster With Cilium CNI Plugin



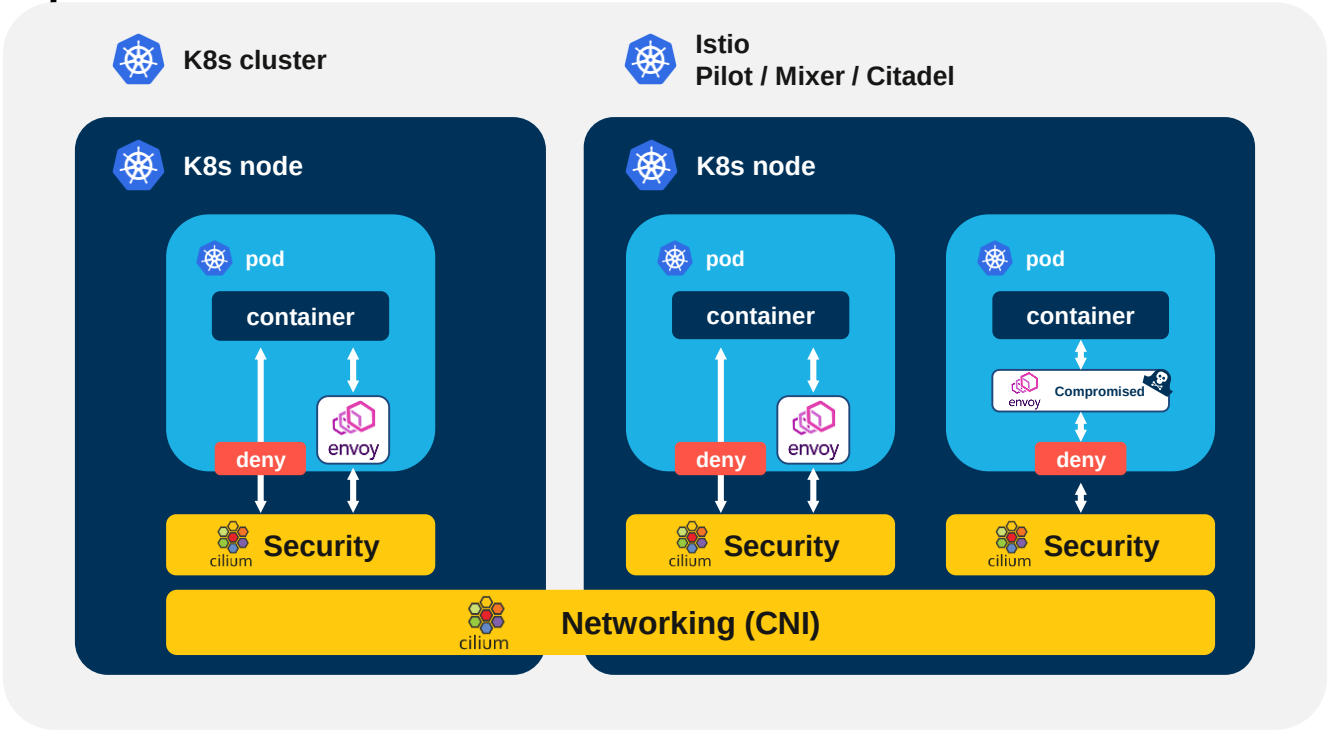
Transparent Sidecar Injection With Cilium



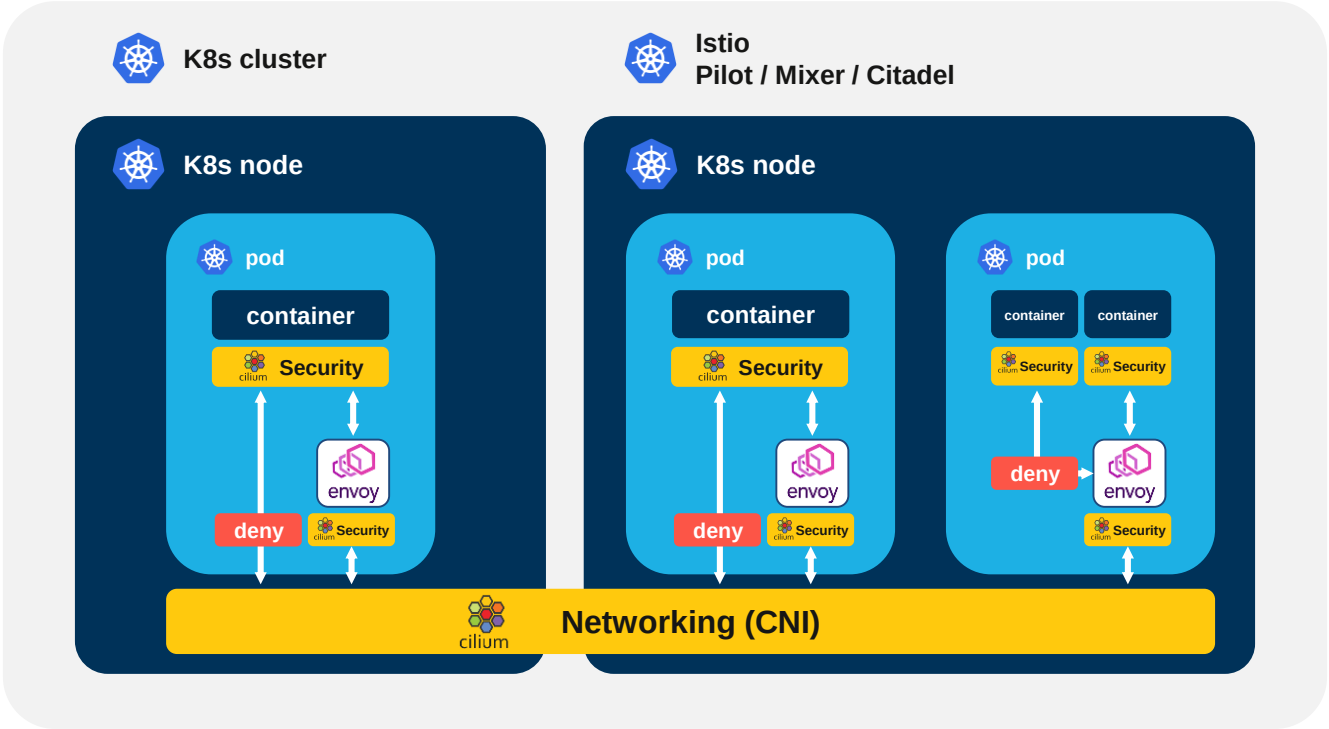
Service Proxy Performance Improvements



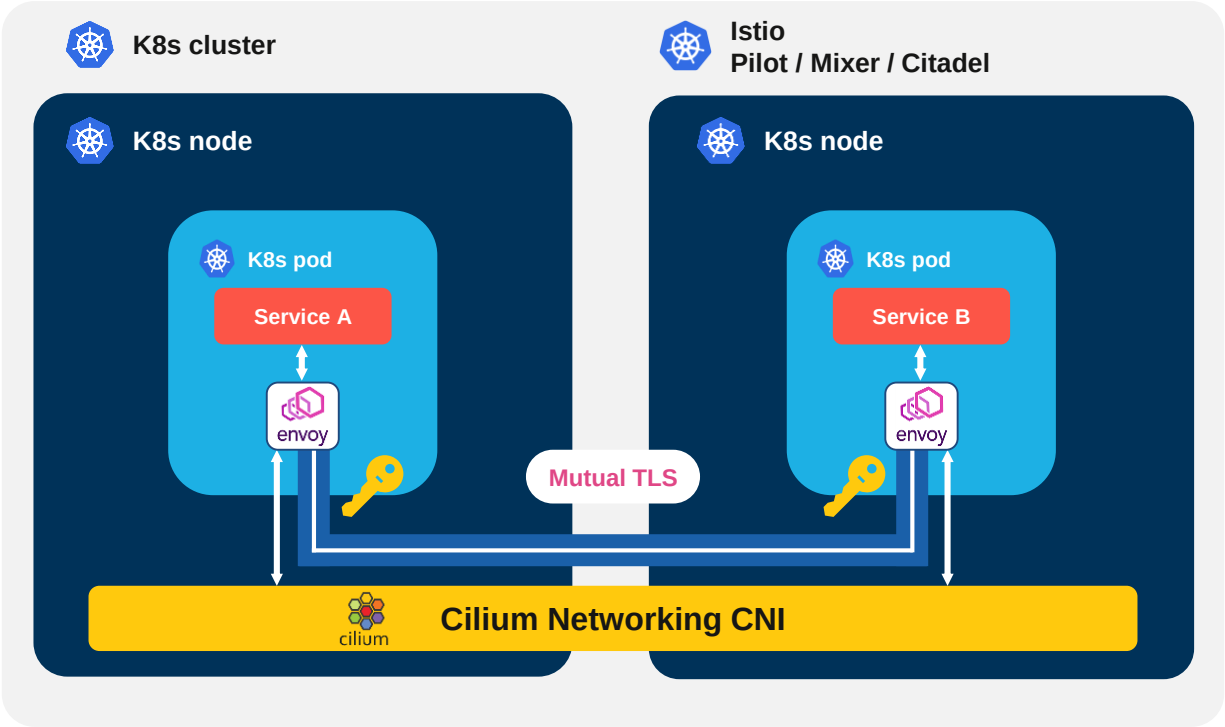
Cilium Protects Unsupported Protocols and Compromised Sidecars



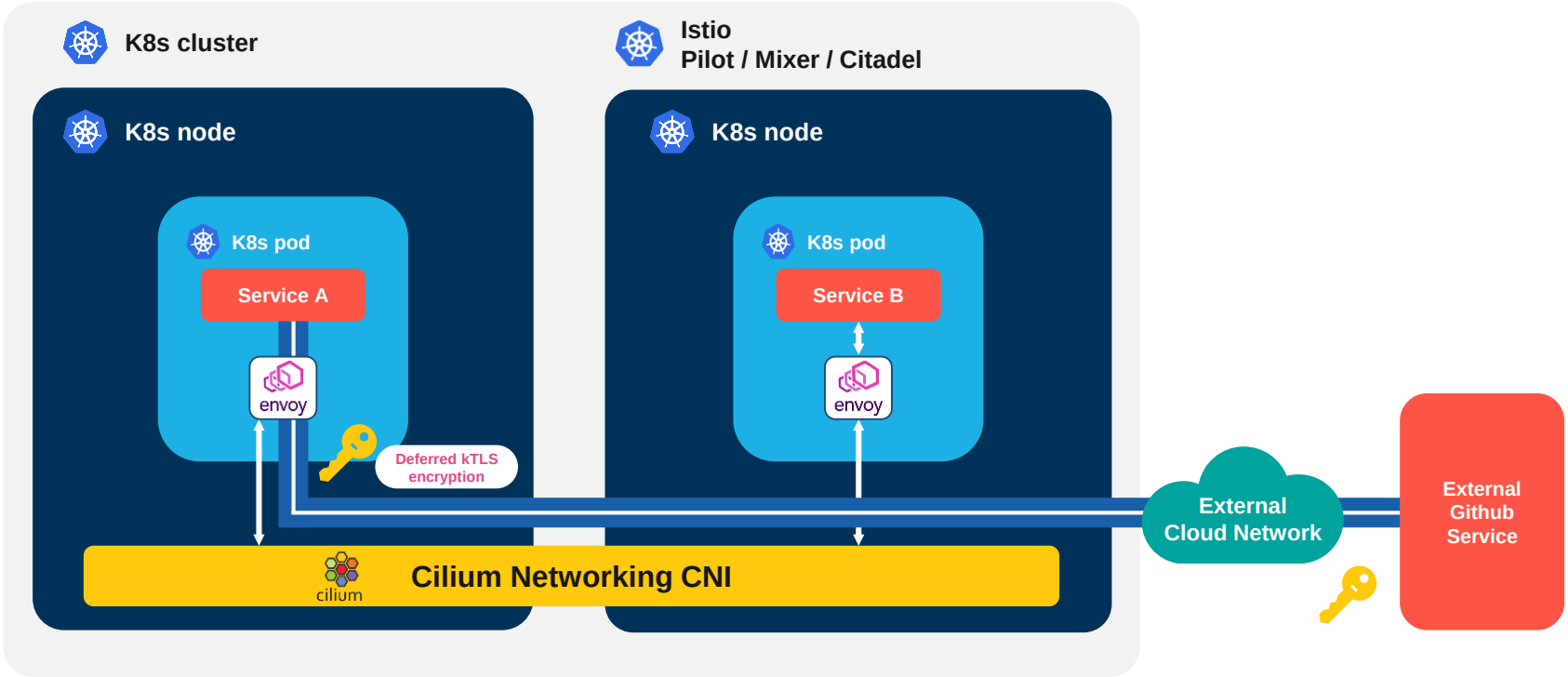
Cilium Secures Multi-Container Pods



Istio – Service-to-Service With Mutual TLS



Istio – Service-To-External Entity With Deferred Ktls



Why Istio Matters?

Istio is stable, has valuable features, and allows for a more granular level security.

It helps to connect

- Control various API calls between services & the traffic flow between them
- Connect microservices
- Secure microservices
- Provides security by default – no modifications required in app code & infrastructure
- Ultra defense: provides multiple layers of security by integrating with another security system
- Allows traffic encryption, helps against MITM attacks
- Control microservices
- Applies enforcement policies
- Observe services microservices
- Provides auto-tracing, logging, and monitoring of all microservices, visualizes what's happening under the hood

And with Cilium we get added performance and security features.

Conclusion

Service mesh is an excellent infrastructure addition for a microservices architecture.

Service Mesh-like Istio enables client-side load balancing and performs all the functions at the application layer.

Istio detects when services fail, slow down, or only partially succeed. And it passes all telemetry info to dashboards such as **Grafana**, simplifying the troubleshooting and tracking down of root problem causes.

Istio enhances the security layer for all communications happening in the service mesh.

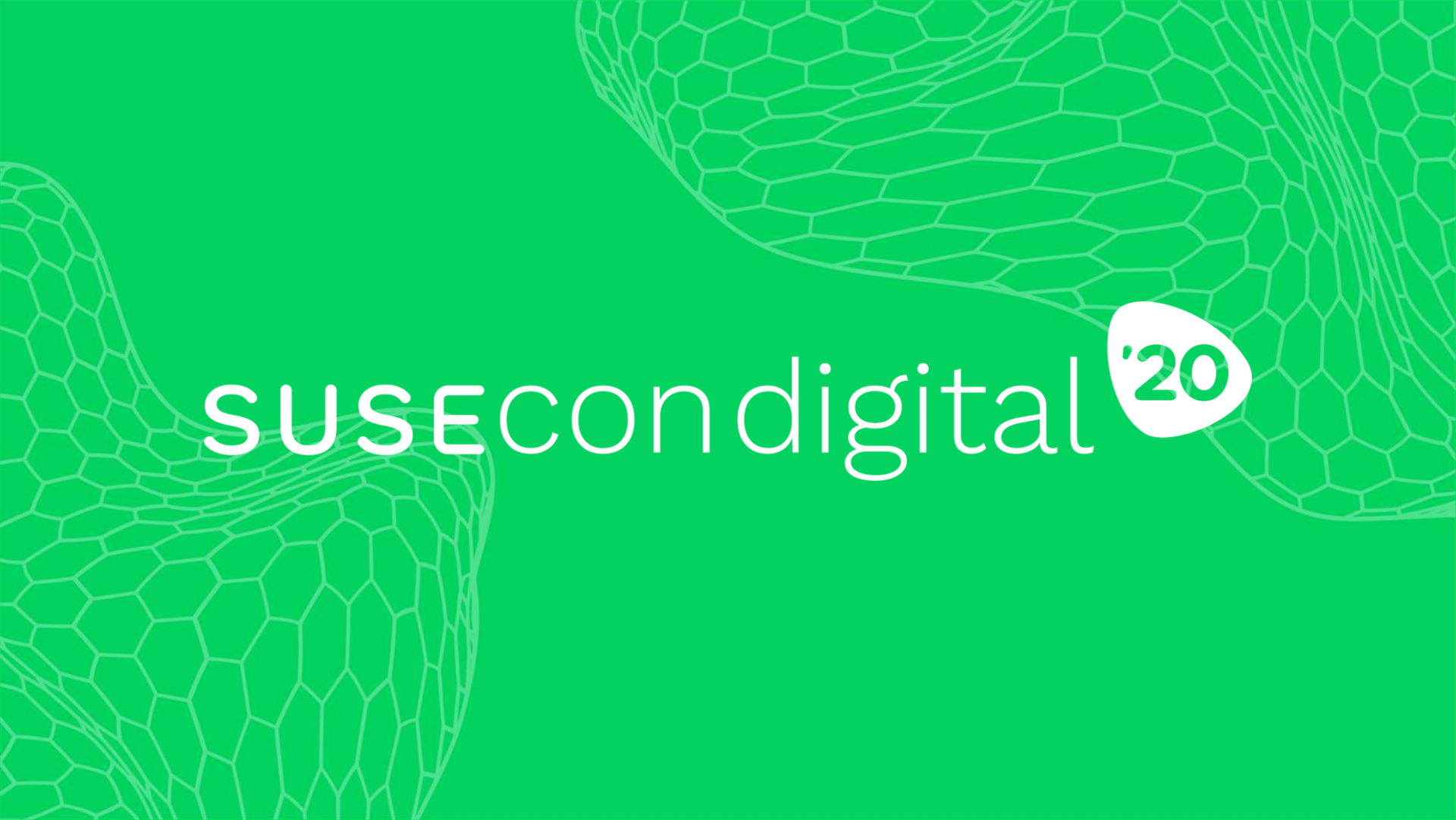
Istio provides the tools needed to run microservices architectures.

Istio provides resilience, routing, and observability.

Q&A
Please Submit Your Questions

General Disclaimer

This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. SUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for SUSE products remains at the sole discretion of SUSE. Further, SUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All SUSE marks referenced in this presentation are trademarks or registered trademarks of SUSE, LLC, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.


SUSEcon digital '20