SERVICE MESHES

Craig Dillon
MSc DevOps
x00205790@mytudublin.ie

WHAT ARE SERVICE MESHES?

"Service meshes provide policy-based, network services for network-connected workloads by enforcing desired behavior of the network in the face of constantly changing conditions and topology."

Source: Calcote & Butcher (2019)

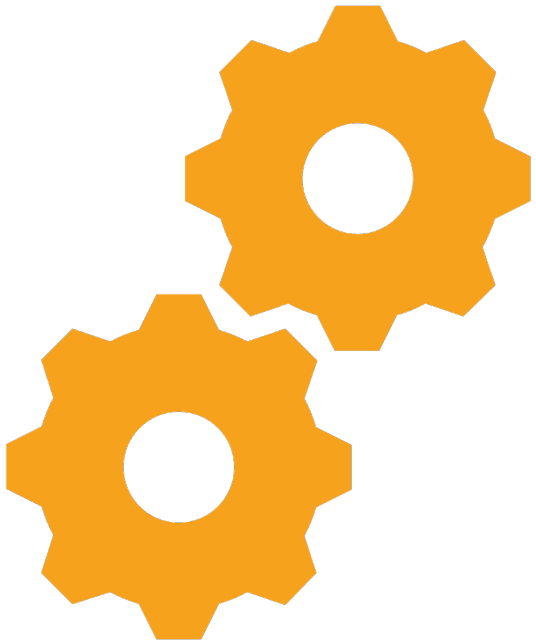
WHAT DOES THAT REALLY MEAN?

- Instead of adding logic to application code to handle communication, metrics and security functions between microservices, offload these workloads to a service mesh for management.
- This ensures a consistent internal network configuration across microservices, reducing development overhead and providing resiliency.

WHY SERVICE MESHES?

- First there were monoliths
- Eventually evolved into microservices where applications are now broken down into several microservice components
- Presented a new set of problems with service communication
- A service mesh is a means to allow reliable, secure and observable communication between these components

Sources: Khatri & Khatri (2020), Duarte Maia & Figueiredo Correia (2022)



PRINCIPLES OF SERVICE MESHES

- Observability
- Routing
- Automatic scaling
- Separation of duties
- Trust
- Automatic service registration and discovery
- Resilience

Source: Li et al (2019), Khatri & Khatri (2020)

COMPONENTS OF A SERVICE MESH

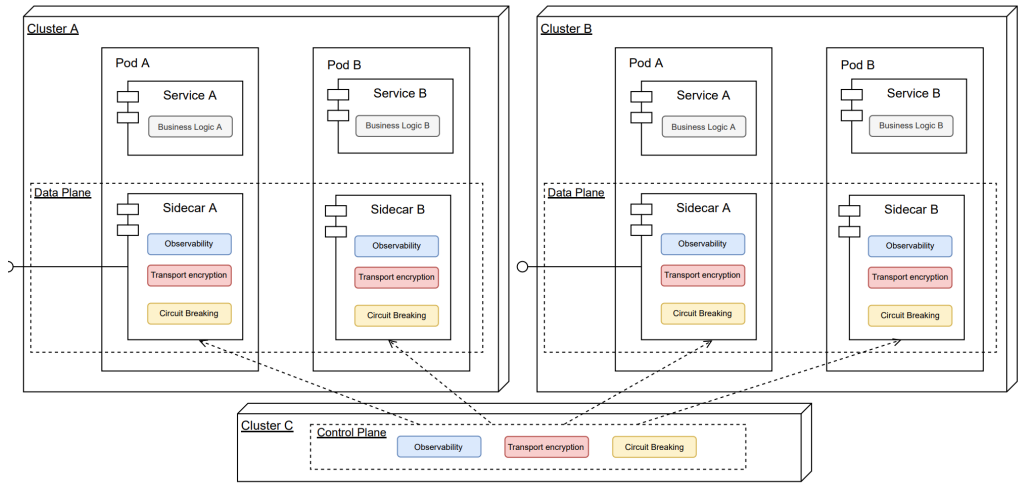
- Control plane: The Control Plane handles service discovery, metric collection, certificate management and configuration of the Data Plane.
- Data plane: Handles ingress and egress traffic for microservices, message processing, encryption enforcement and metric collection.

SUBCOMPONENTS OF A SERVICE MESH

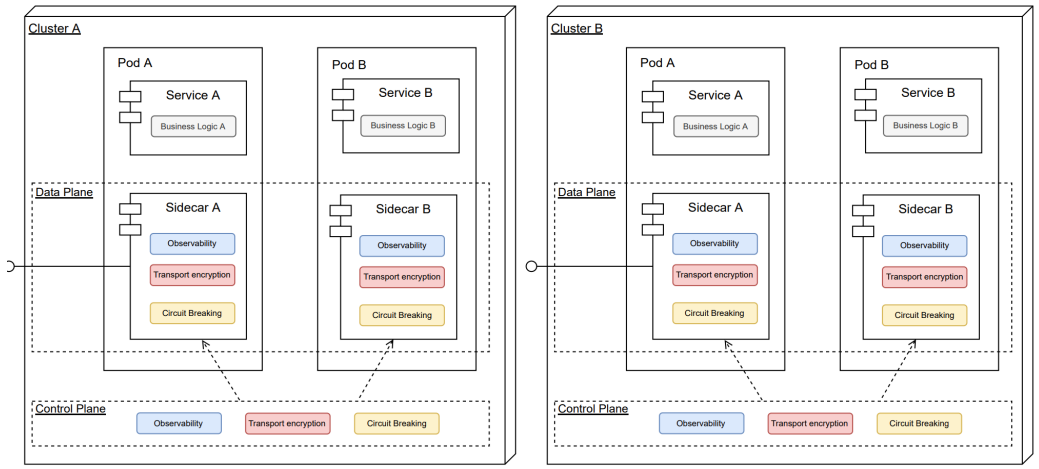
- Data Plane:
  - Proxy: This is usually in the form of a sidecar container running alongside the microservice in each pod.
- Control Plane:
  - Traffic manager – Performs service discovery and when a change is detected notifies proxies to modify routing as needed.
  - Security component – Encrypts traffic between microservices, often using *mutual Transport Layer Security* (mTLS) in a transparent manner without making changes to the application.
  - Configuration component – Responsible for validating and distributing configurations to components of the service mesh.

Source: Koshel et al (2021)

ARCHITECTURE OF SERVICE MESHES

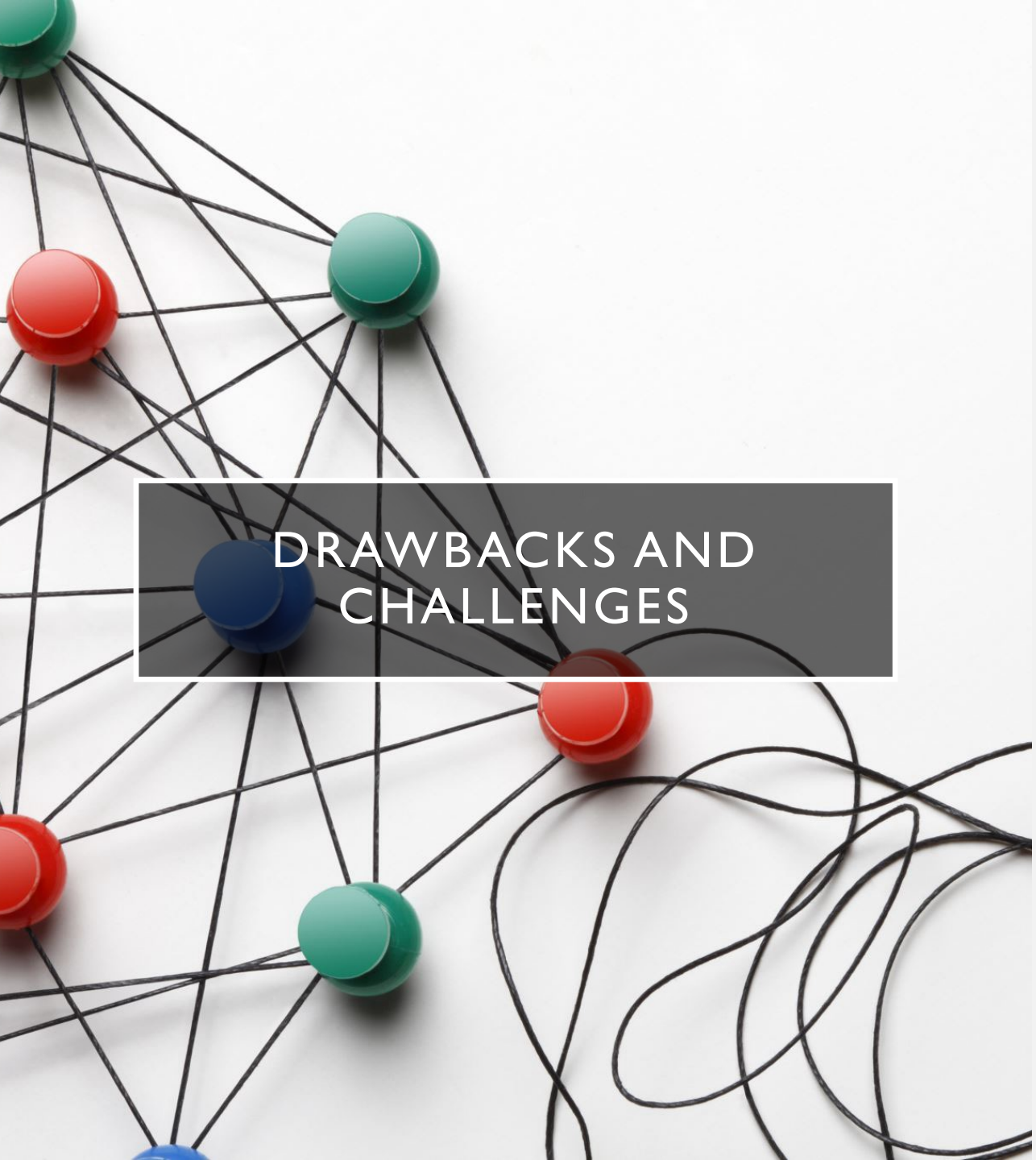


Service Mesh with Sidecar



Multi Cluster Service Mesh

Source: Duarte Maia & Figueiredo Correia (2022)



DRAWBACKS AND CHALLENGES

- Increased complexity
- Performance overhead
- Operational overhead
- Vendor lock-in
- Learning curve

- Largest fashion E-commerce company in Japan
- Initially deployed as an on-premises monolithic platform in 2014
- In 2018 moved to a hybrid K8s platform with an API Gateway added in 2020 which added development overheads
- 2021 saw the introduction of Istio as a service mesh

Source: Kawasaki (2022)

- Xbox Cloud launched in 2019 and used an in-house developed mTLS (mutual TLS) to ensure secure communication between services
- As the service grew the development and maintenance overhead also grew
- Moved to Linkerd as the service mesh solution
- Kustomize was used to configure Linkerd and other components
- Service is built on AKS (Azure Kubernetes Service) and Linkerd works in conjunction with Flagger to allow canary deployments
- Massively reduced engineering time, improved reliability, saved thousands per month and reduced latency by 100ms

Source: Wodajie & Voss (2022)

EMERGING TRENDS

- Moving service mesh functions to eBPF, Cilium is currently the leading service mesh applying this method
- Kuma is a solution that is designed to be environment agnostic and supports multi-zone, multi-cluster and multi-cloud deployments
- Traefik Mesh is designed to be a lightweight, non-invasive service mesh meaning it runs a dedicated pod on each node to handle routing. Services are added manually rather than via service discovery.

COMPARISON OF SERVICE MESHES

				
Sidecar Proxy	**Yes**	**Yes**	**Yes**	**Optional**
Per node agent	**No**	**No**	**Yes**	**Yes**
Multicluster support	**Yes**	**Yes**	**Yes**	**Yes**
Deployment method	**Helm**	**Helm**	**Helm**	**Helm**
Ingress controller	**Any**	**Istio ingress or Istio gateway**	**Envoy & ambassador**	**Cilium Ingress, compatible with others**
Other considerations			**Requires Consul server for quorum operations**	**Is a Kubernetes network component rather than service**

CRITICAL SUMMARY

- Streamlined communication between microservices
- Does not replace operational overhead but potentially reduces it
- Not all service meshes are created equal
- Rapidly evolving ecosystem
- Overall provides a holistic means of managing networking between microservices

REFERENCES

- Calcote, L. & Butcher, Z. (2019), Istio: Up and running: Using a service mesh to connect, secure, control, and observe, O'Reilly Media.
- Duarte Maia, J. T. & Figueiredo Correia, F. (2022), Service mesh patterns, in 'Proceedings of the 27th European Conference on Pattern Languages of Programs', pp. 1–12.
- Kawasaki, Y. (2022), 'Accelerating zozotown modernization with istio'. URL: https://istio.io/latest/about/case-studies/zozo/ (accessed 18/03/2024)
- Khatri, A. & Khatri, V. (2020), Mastering Service Mesh: Enhance, secure, and observe cloud-native applications with Istio, Linkerd, and Consul, Packt Publishing Ltd.
- Koschel, A., Bertram, M., Bischof, R., Schulze, K., Schaaf, M. & Astrova, I. (2021), A look at service meshes, in '2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA)', IEEE, pp. 1–8.
- Li, W., Lemieux, Y., Gao, J., Zhao, Z. & Han, Y. (2019), Service mesh: Challenges, state of the art, and future research opportunities, in '2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)', IEEE, pp. 122–1225.
- Wodajie, A. & Voss, C. (2022), 'Service mesh at scale: How xbox cloud gaming secures 22k pods with linkerd'. URL: https://www.cncf.io/blog/2022/05/10/service-mesh-at-scale-how-xbox-cloud-gaming-secures-22k-pods-with-linkerd (accessed 23/03/2024)
- Zhu, X., She, G., Xue, B., Zhang, Y., Zhang, Y., Zou, X. K., Duan, X., He, P., Krishnamurthy, A., Lentz, M. et al. (2022), 'Dissecting service mesh overheads', arXivpreprint arXiv:2207.00592 .3