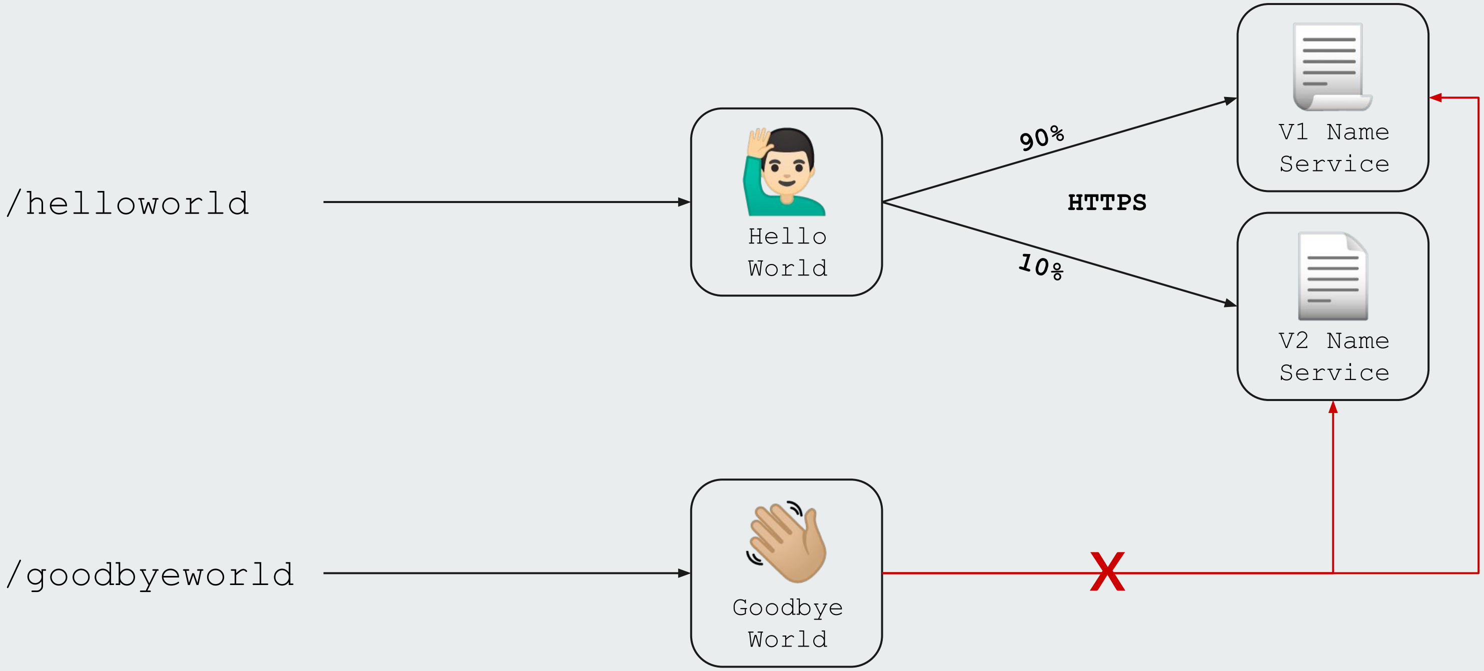
“Hello World”

every developer, always

“Goodbye World”

no developer, ever



/helloworld

Hello World

90%

HTTPS

10%

V1 Name Service

V2 Name Service

/goodbyeworld

Goodbye World

X

Service Meshes as a Solution for Developing Technology Independent Microservices

Kilian Dangendorf, Eike Kirch, Christopher Rust, Manuel Ottlik

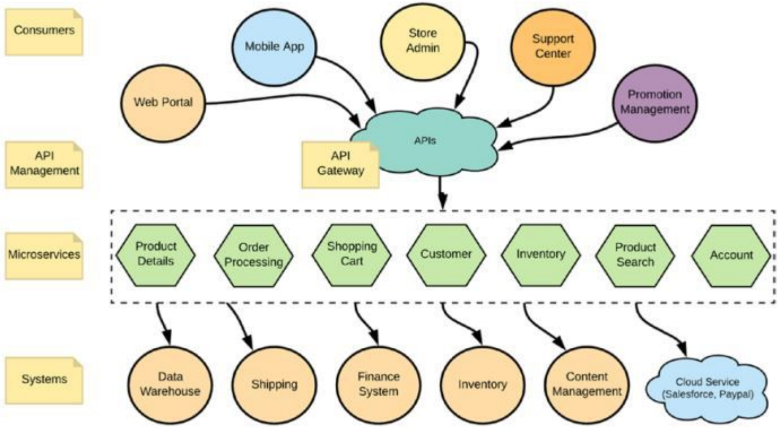
Agenda

1. Introduction
2. Microservices and their Challenges
3. Overview of Base Technologies
4. Characteristics of Service Meshes
5. Available Solutions
6. Prototypical Service Mesh Implementation
7. Conclusion & Reflection

Microservices and their Challenges

Characteristics

- high autonomy
- self-contained
- business-oriented
- small & independent
- reused in several business processes



A more complex answer to a more complex problem

Horizontal Scaling

provides more flexibility in comparison to monolithic applications

enables demand oriented scaling of only the highly demanded services

Independent Development

small teams can focus on developing a service in their preferred language with little side constraints

testing microservices often is easier due to less side constraints and well defined APIs between services

Independent Deployment

updates of services can be performed more frequently allowing a shorter time to market

reduces the need for release management compared to monolithic applications

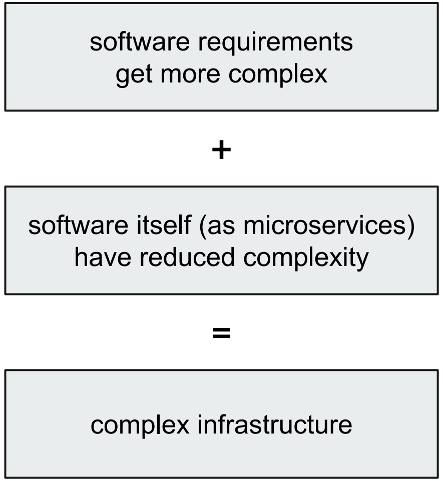
Shift of Complexity towards Infrastructure

„A microservice architecture shifts around complexity.

Instead of a single complex system, you have a bunch of

simple services with complex interactions.“

Vinay Sahni



software requirements get more complex

+

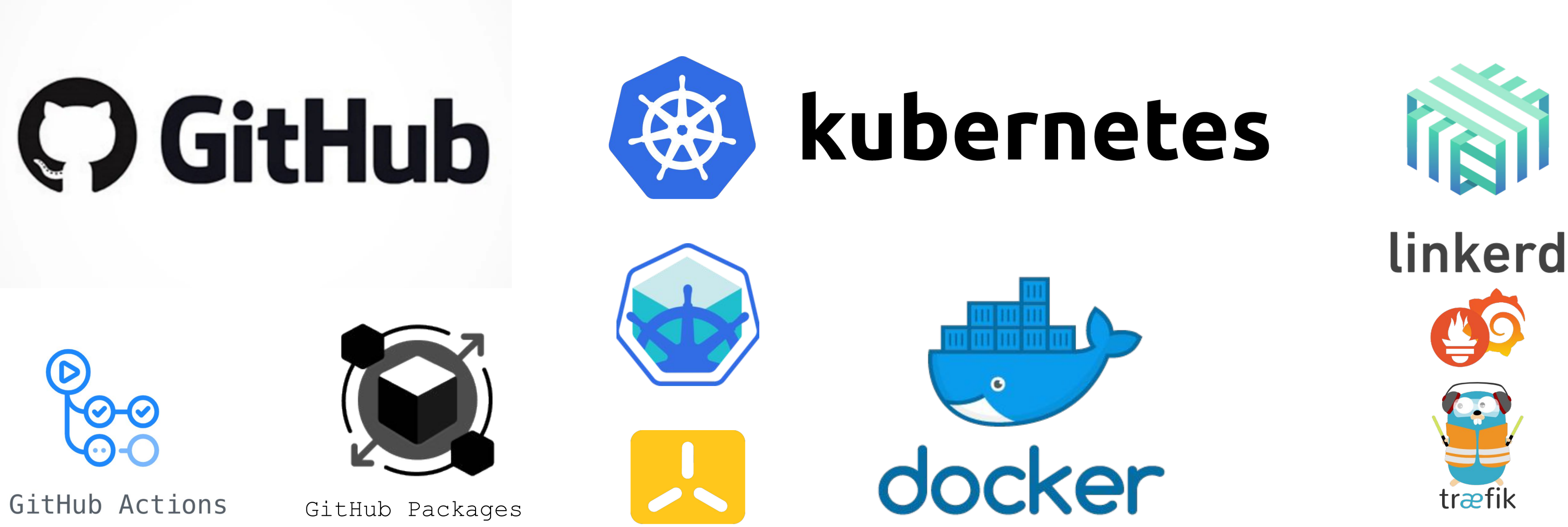
software itself (as microservices) have reduced complexity

=

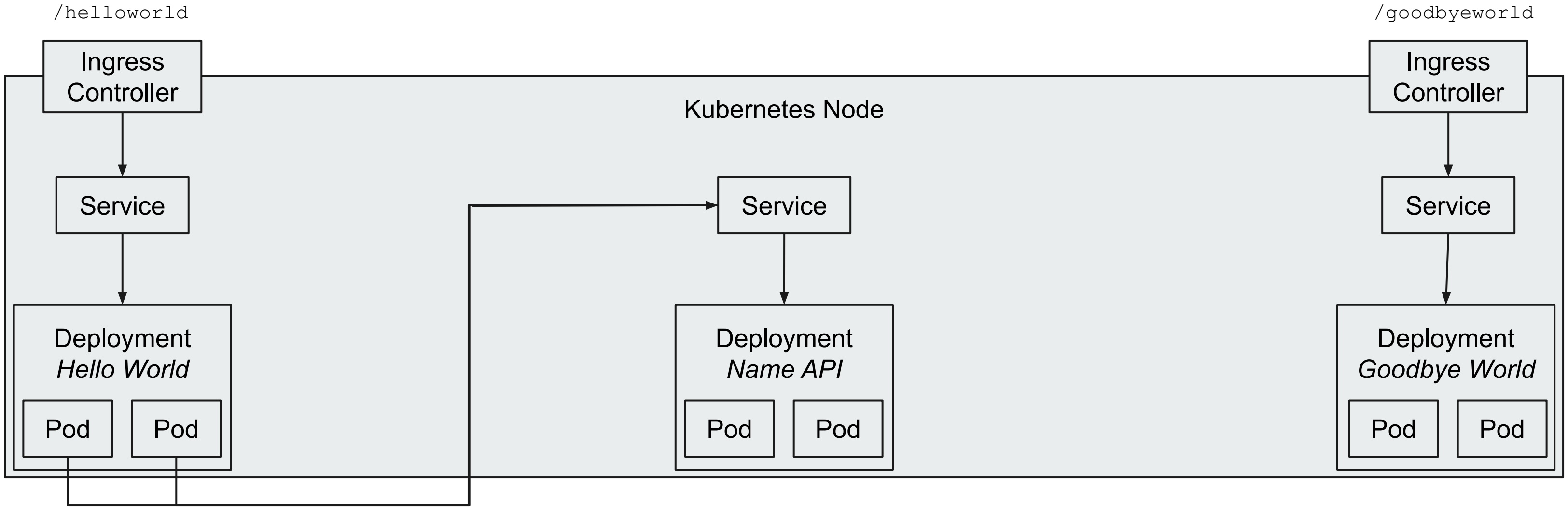
complex infrastructure

Overview of Base Technologies

Tech Stack

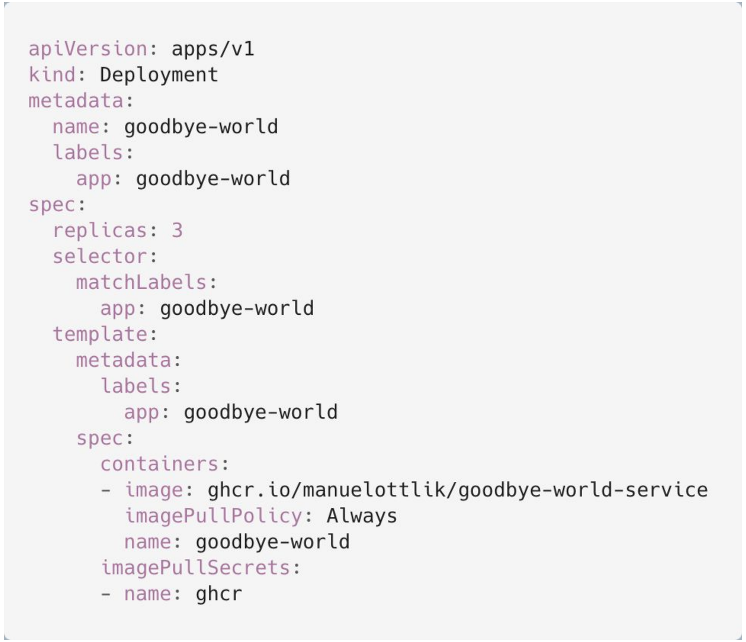

GitHub
kubernetes
linkerd
GitHub Actions
GitHub Packages
docker
træfik

Basic Kubernetes Resources



Deployment Resource Example

- declarative approach to define infrastructure
- ensures a specific state in the cluster
- encapsulates logic for rolling updates
- enables automatic scaling

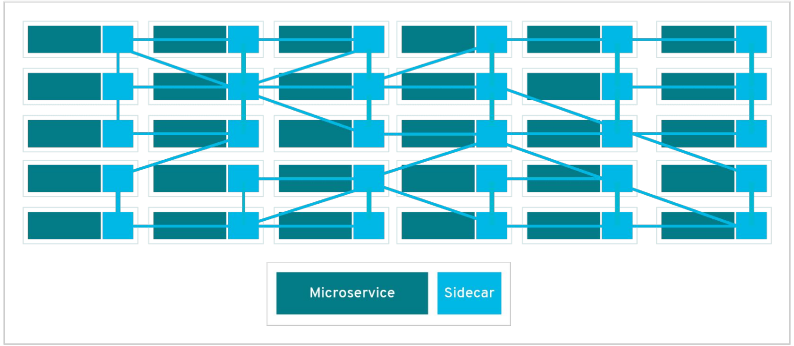


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: goodbye-world
  labels:
    app: goodbye-world
spec:
  replicas: 3
  selector:
    matchLabels:
      app: goodbye-world
  template:
    metadata:
      labels:
        app: goodbye-world
    spec:
      containers:
      - image: ghcr.io/manuelottlik/goodbye-world-service
        imagePullPolicy: Always
        name: goodbye-world
      imagePullSecrets:
      - name: ghcr
```

Characteristics of Service Meshes

Definition

“A service mesh is a dedicated infrastructure layer for handling service-to-service communication. It’s responsible for the reliable delivery of requests through the complex topology of services that comprise a modern, cloud native application. In practice, the service mesh is typically implemented as an array of lightweight network proxies that are deployed alongside application code, without the application needing to be aware.”



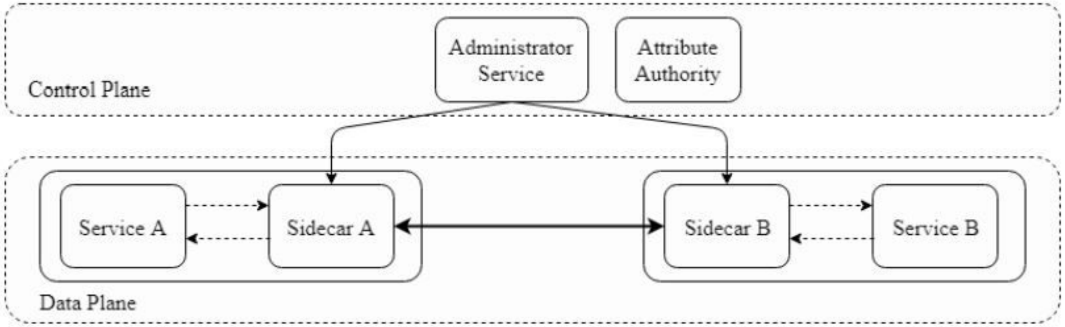
Sidecar-Architecture

Lightweight proxy container is added inside every pod.

These proxies control the entire network communication.

All proxies are controlled by a central instance.

The central instance applies the defined rules and features.



Central Tasks

Service Discovery

enables communication between services to be more intuitive to developers

Load Balancing

skillfully distributes requests among replicas in high demand situations

Fault Tolerance

takes health state of service into account when redirecting requests to other services

Traffic Monitoring

records and displays all communication between services for debugging & logging purposes

Circuit Breaking

prevents overloading of faulty resources due to connection problems or complex operations

Access Control

restricts communication between services to apply the least privilege principle

Advantages & Drawbacks

- + freedom of choice for programming languages & frameworks
- + focus on business logic
- + infrastructure tasks get less complicate
- + extends declarative approach of infrastructure
- + out of the box monitoring & visualization

- - increased complexity
- - increased traffic and resources
- - new technology that might not be mature enough already

Available Solutions

Overview of available options

Istio

træfik
mesh

LINKERD

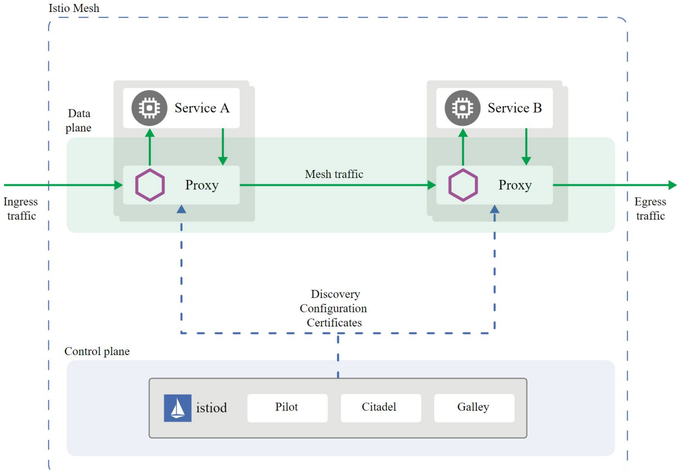
HashiCorp
Consul

Istio vs. Linkerd: Characteristics

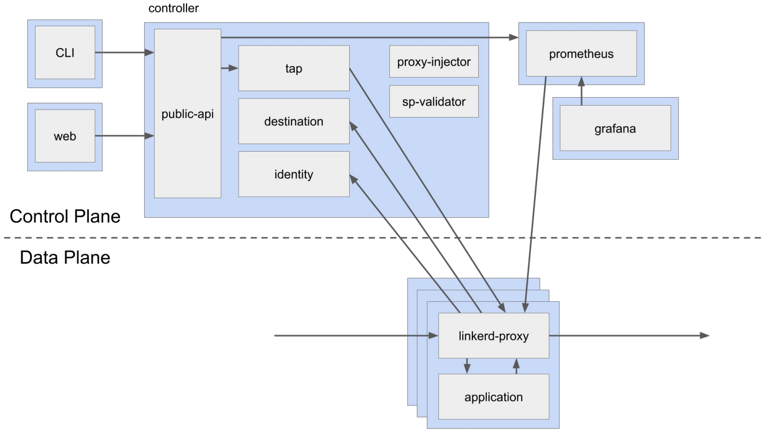
- initiated by Lyft, IBM, Google
- designed for Kubernetes & VMs
- more features with more options
- more advanced yet more complex
- 26.2k stars on GitHub
- very detailed documentation

- initiated by Bouyant & CNCF
- designed for Kubernetes only
- super lightweight design
- circuit breaking still in development
- 6.6k stars on GitHub
- good / acceptable documentation

Istio vs. Linkerd: Architecture



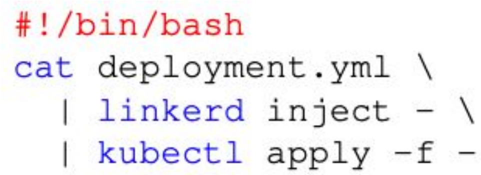
Istio



LINKERD

Why Linkerd?

- easy to setup
- super lightweight approach
- ideal for proof of concepts, local experiments etc
- supports other lightweight kubernetes concepts
- recommends minikube & traefik

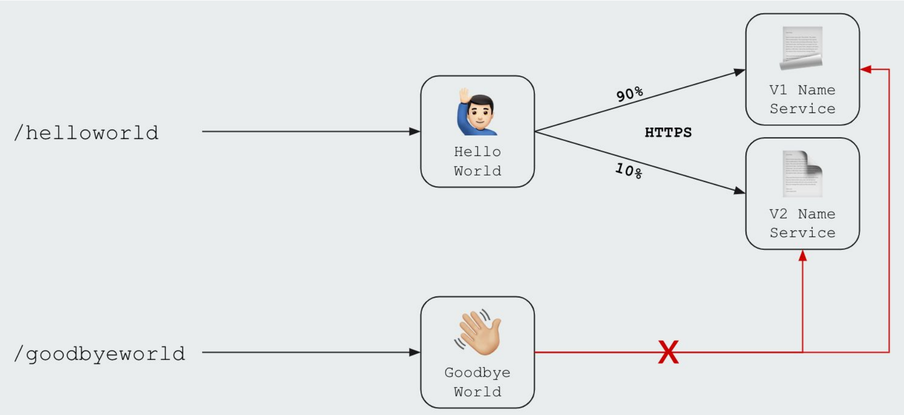


```
#!/bin/bash
cat deployment.yml \
   | linkerd inject - \
   | kubectl apply -f -
```

Prototypical Service Mesh Implementation

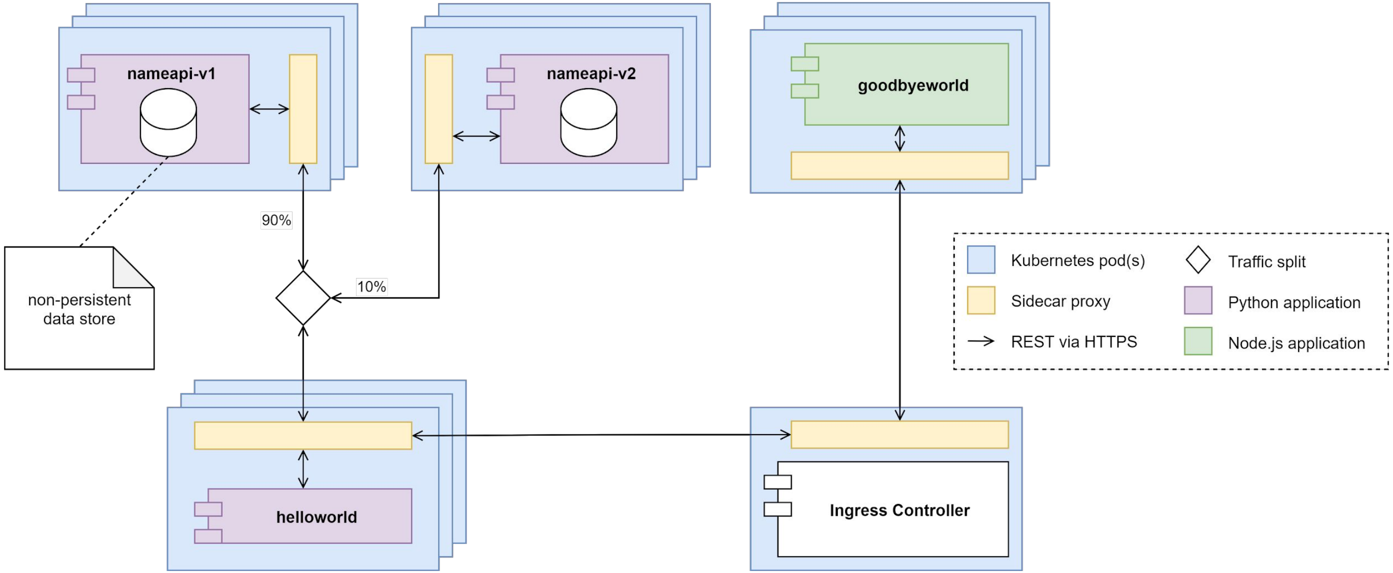
Showcases

1. Cluster without Service Mesh (no TLS)
2. Add Services to Service Mesh (TLS)
3. Load Balancing between Replicas
4. Add 90/10 Traffic Split



Monitoring along the way...

Prototype Architecture



It’s time for a demo!

Conclusion & Reflection

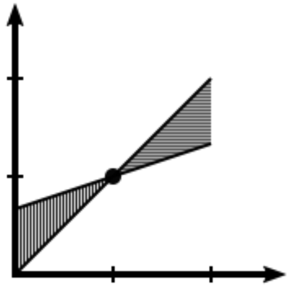
Limitations of Linkerd

- builds upon & deeply integrates kubernetes features
- still in early phase of development
- inferior to Istio for production
- documentation is partly outdated
- lightweight not only in terms of setup but features
- many features require additional plugins

but: TLS, metrics, monitoring including Prometheus & Grafana come out of the box

Break Even Point

- huge tech stack to setup & learn
- prototype application consumes 4GB of RAM
- four people and two weeks only brought us to the surface



but: effort will pay off in production clusters in need of monitoring & metrics

Thank you for your time and attention!

Any questions?

Visit https://github.com/manuelottlik/hsh-favs to see how everything works.

Image Sources

- Indrasiri, Kasun., and Prabath. Siriwardena. “Microservices for the Enterprise”: Designing, Developing, and Deploying / by Kasun Indrasiri, Prabath Siriwardena. 1st ed. 2018., Apress, 2018, p. 7
- https://www.redhat.com/de/topics/microservices/what-is-a-service-mesh (Last access: 2020/02/24, 2 pm)
- https://linkerd.io/2/reference/architecture/ (Last access: 2021/02/18, 6 pm)
- https://istio.io/latest/docs/ops/deployment/architecture/ (Last access: 2021/02/18, 6 pm)
- https://github.com (Last access: 2021/02/24, 9 pm)
- https://kubernetes.io (Last access: 2021/02/24, 9 pm)
- https://k3s.io (Last access: 2021/02/24, 9 pm)
- https://www.docker.com (Last access: 2021/02/24, 9 pm)
- https://grafana.com (Last access: 2021/02/24, 9 pm)
- https://traefik.io (Last access: 2021/02/24, 9 pm)

Paper Sources

- P. Giessler, “Domänengetriebener Entwurf von ressourcenorientierten Microservices”, Karlsruher Institut für Technologie (KIT), 2018
- Spring Cloud https://spring.io/projects/spring-cloud (Last access: 2021/02/11, 11 am)
- R. Chen, S. Li and Z. Li, ”From Monolith to Microservices: A Dataflow-Driven Approach,” 2017 24th Asia-Pacific Software Engineering Conference (APSEC), Nanjing, 2017, p. 473
- W. Li, Y. Lemieux, J. Gao, Z. Zhao and Y. Han, ”Service Mesh: Challenges, State of the Art, and Future Research Opportunities,” 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE), San Francisco East Bay, CA, USA, 2019, pp. 122-1225
- Was ist ein Service Mesh? - https://www.redhat.com/de/topics/microservices/what-is-a-service-mesh (Last access: 2021/02/11, 12 am)
- Indrasiri, Kasun., and Prabath. Siriwardena. “Microservices for the Enterprise”: Designing, Developing, and Deploying / by Kasun Indrasiri, Prabath Siriwardena. 1st ed. 2018., Apress, 2018
- K. Y. Ponomarev, “Attribute-Based Access Control in Service Mesh,” 2019 Dynamics of Systems, Mechanisms and Machines (Dynamics), Omsk, Russia, 2019, pp. 1-4
- linkerd/linkerd2 @ GitHub - https://github.com/linkerd/linkerd2 (Last access: 2021/02/17, 11 am)
- istio/istio @ GitHub - https://github.com/istio/istio (Last access: 2021/02/17, 11 am)
- Kubernetes Service Mesh: A Comparison of Istio, Linkerd and Consul - https://platform9.com/blog/kubernetes-service-mesh-a-comparison-of-istio-linkerd-and-consul/ (Last access: 2021/02/17, 11 am)
- Service Mesh: Einführung & Vergleich von Istio und Linkerd - https://www.predic8.de/service-mesh-microservices.htm (Last access: 2021/02/17, 11 am)
- Circuit breaker support? - https://github.com/linkerd/linkerd2/issues/2846 (Last access: 2021/02/17, 2 pm)
- Microservices - https://martinfowler.com/articles/microservices.html (Last access: 2021/02/17, 4 pm)
- Konzepte — Kubernetes - https://kubernetes.io/de/docs/concepts/ (Last access: 2021/02/18, 3 pm)
- Linkerd: FAQ - https://linkerd.io/2/faq/ (Last access: 2021/02/18, 6 pm)
- Why Linkerd doesn’t use Envoy - https://linkerd.io/2020/12/03/why-linkerd-doesnt-use-envoy/ (Last access: 2021/02/18, 6 pm)
- Linkerd: Architecture - https://linkerd.io/2/reference/architecture/ (Last access: 2021/02/18, 6 pm)
- Linkerd: Get started - https://linkerd.io/2/getting-started/ (Last access: 2021/02/20, 3 pm)
- Istio: Architecture - https://istio.io/latest/docs/ops/deployment/architecture/ (Last access: 2021/02/18, 6 pm)
- Getting Started — Circuit Breaker - https://spring.io/guides/gs/circuit-breaker/ (Last access: 2021/02/20, 5 pm)
- Hystrix EOL - https://github.com/Netflix/Hystrix/blob/master/README.md
- Ingress Traffic - https://linkerd.io/2/tasks/using-ingress (Last access: 2021/02/22, 3 pm)