



INOQ

TECHNOLOGY NIGHT

15.01.2020
INNOQ Technology Night

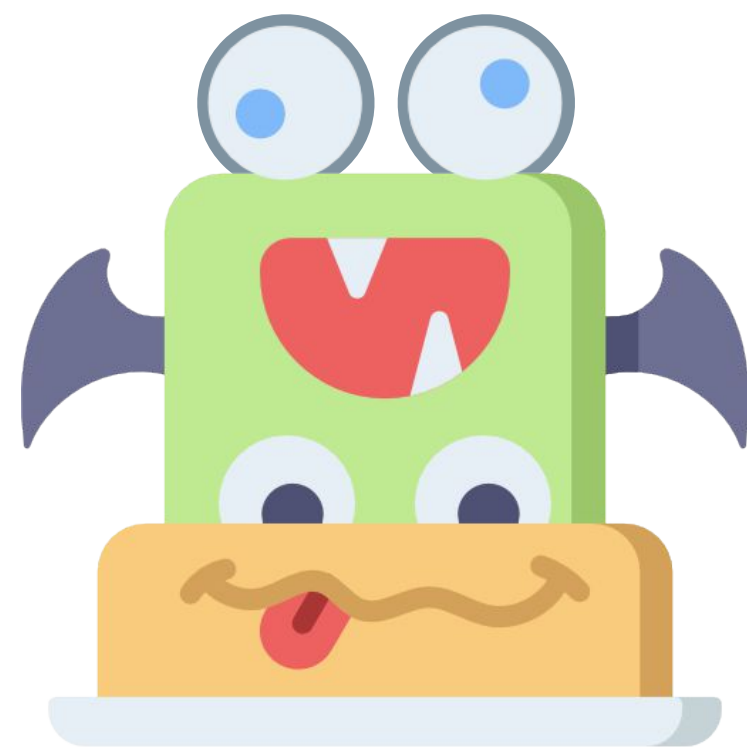


Service Mesh - Was die neue Infrastruktur für Microservices taugt

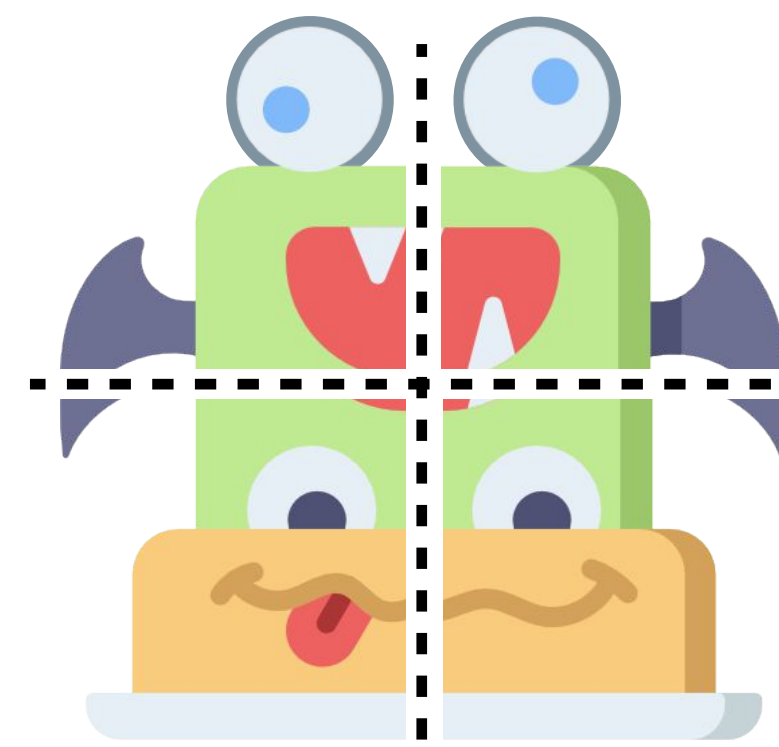
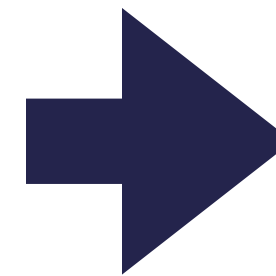


Jörg Müller
Hanna Prinz

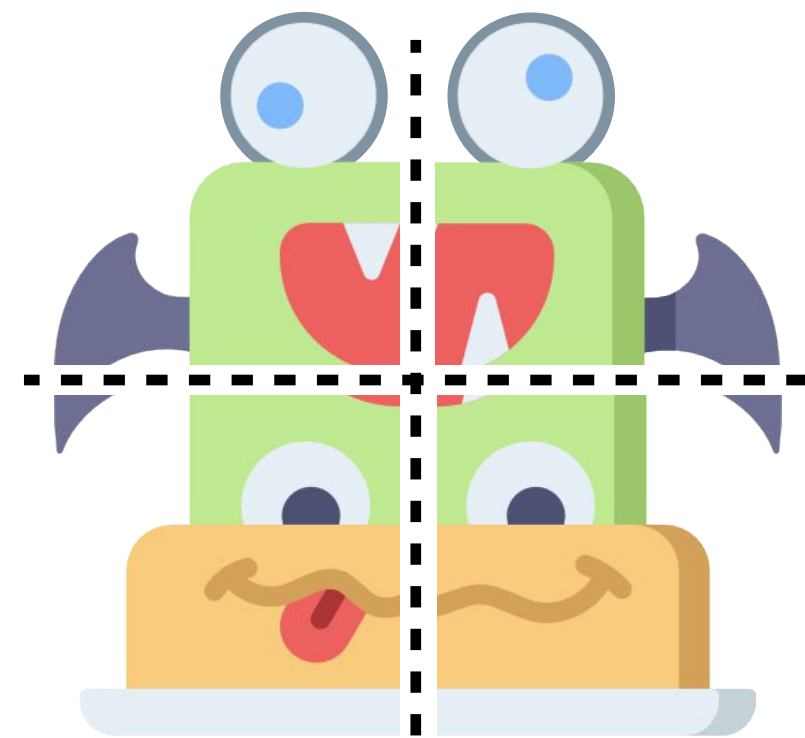
Warum Service Meshes?



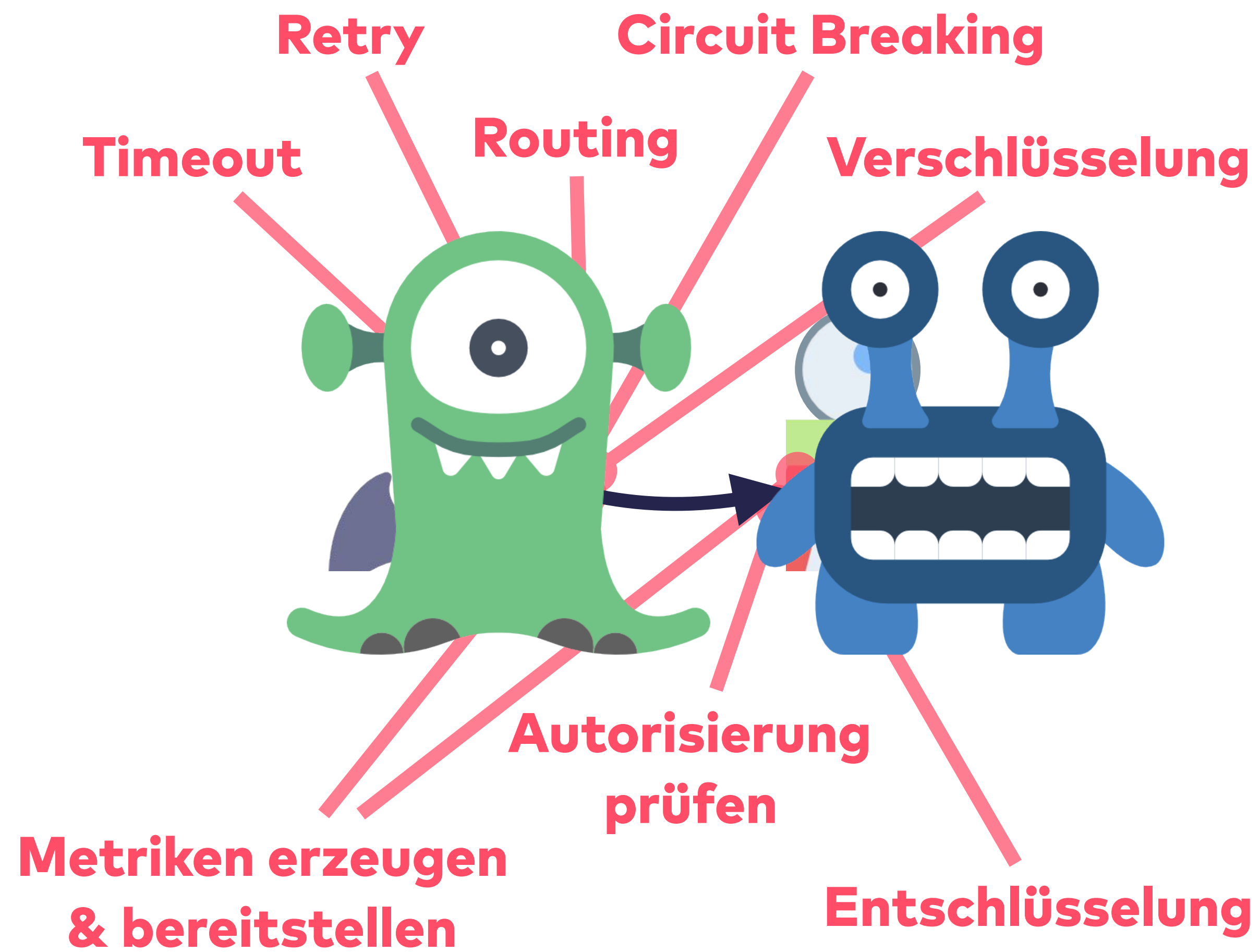
Monolith



Microservices

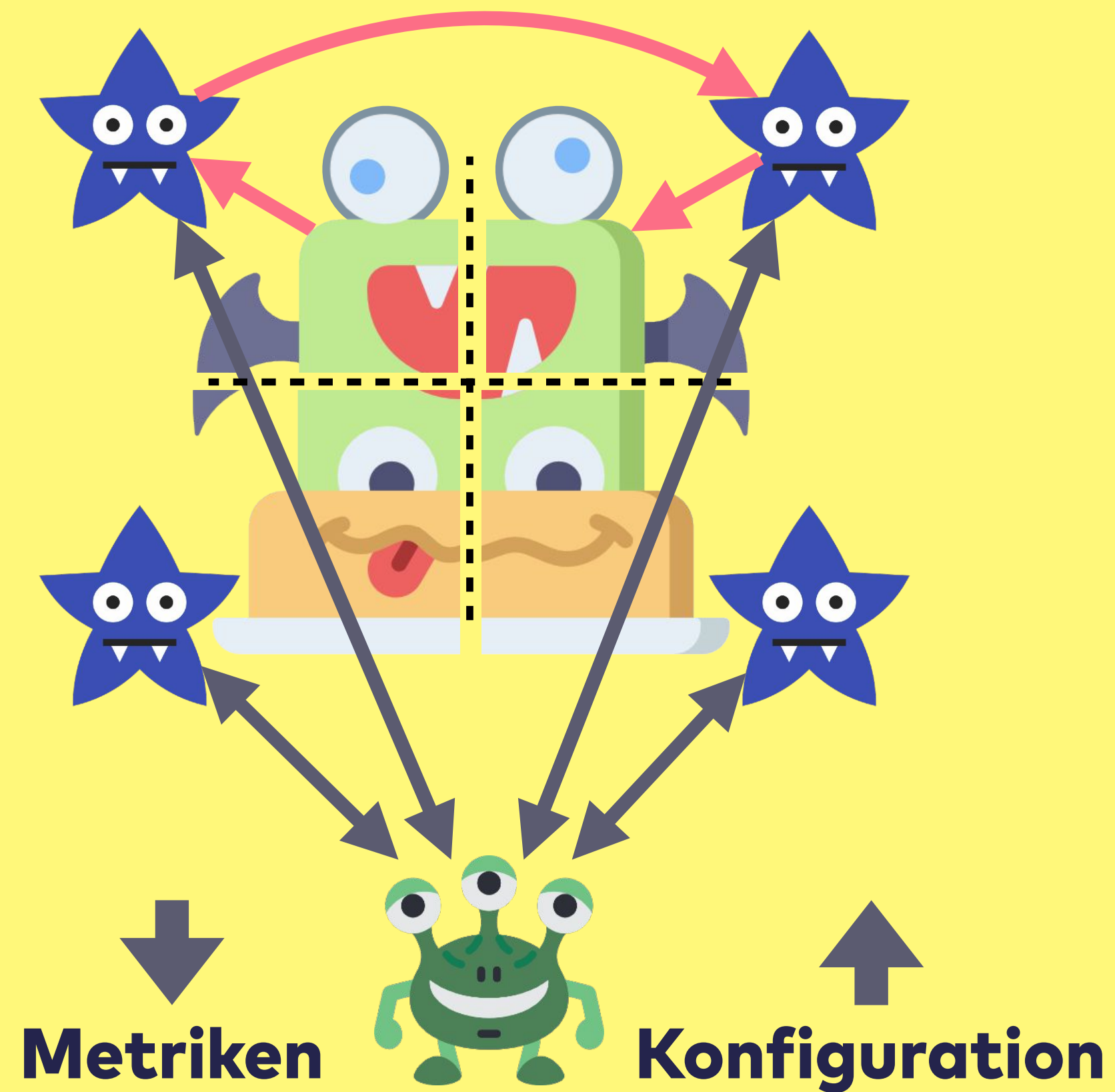
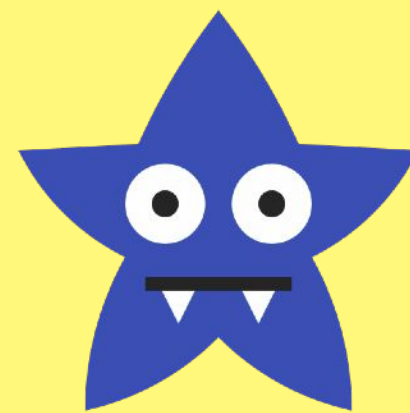


Microservices

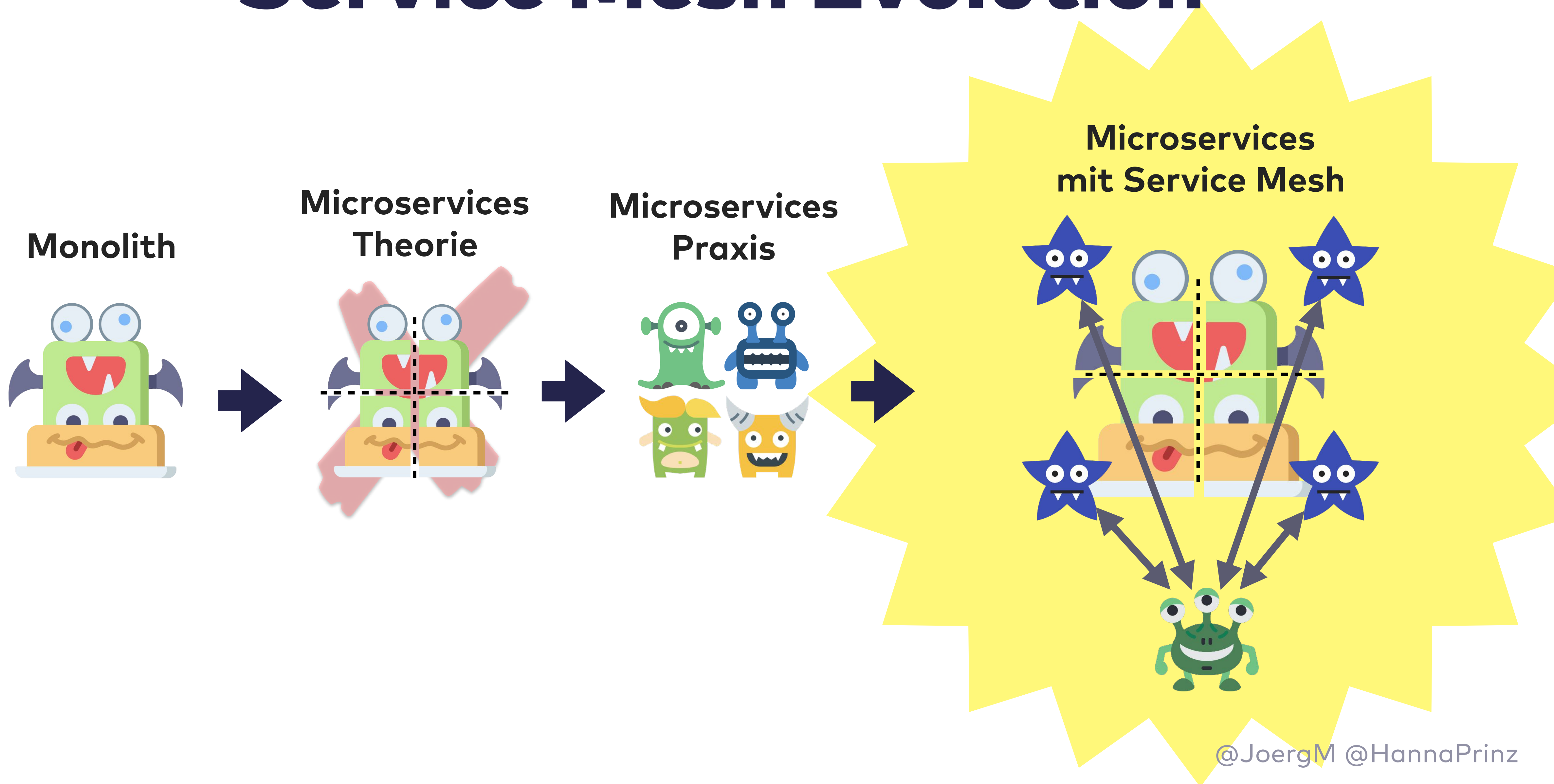


Service Mesh

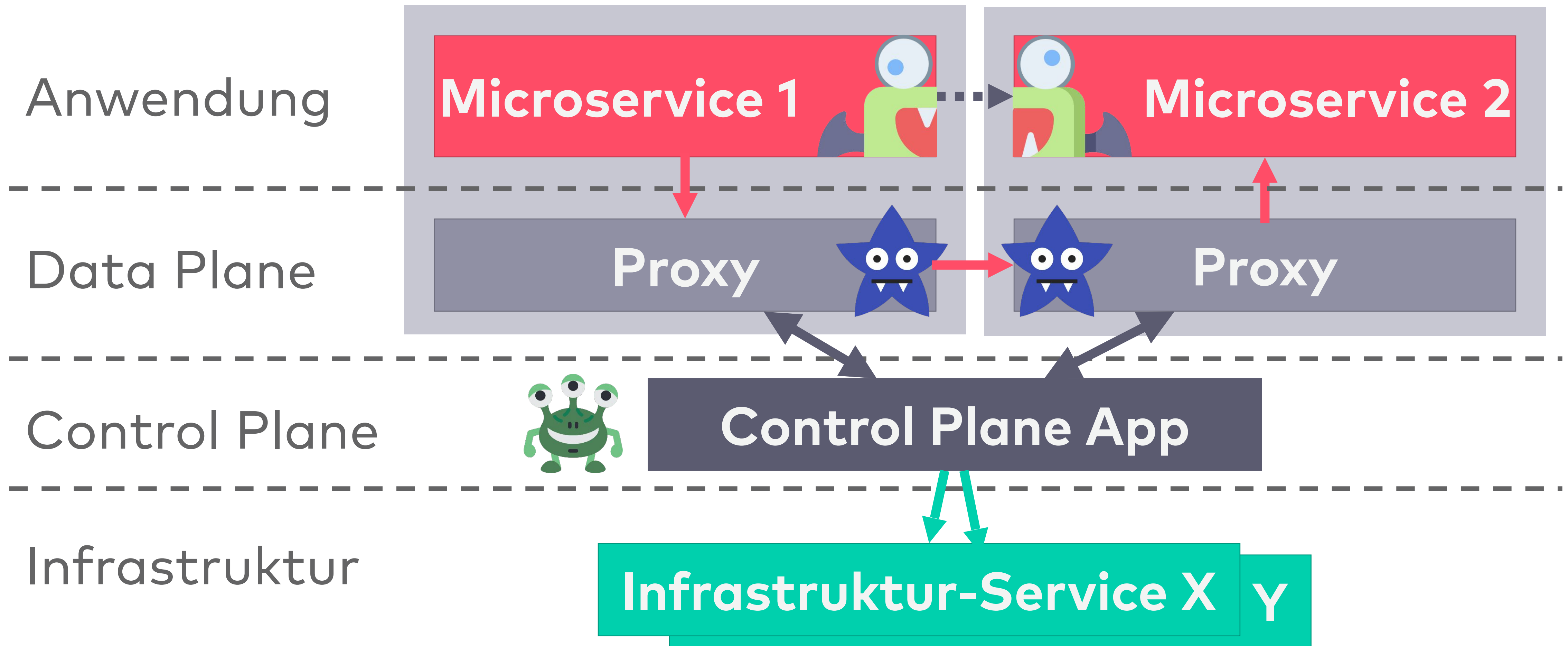
Retry
Timeout
Circuit Breaker
Routing
Verschlüsseln
Entschlüsseln
Autorisierung
Metriken
...



Service Mesh Evolution



Service Mesh Architektur

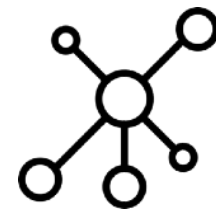


Was kann ein Service Mesh?

Service Mesh Features



Observability



Routing



Resilience



Security



**Beispiele mit dem
Istio Service Mesh**

Monitoring

Ein Service Mesh kann **automatisch**
Daten für alle 4 "**Golden Signals**" liefern:

Latenz

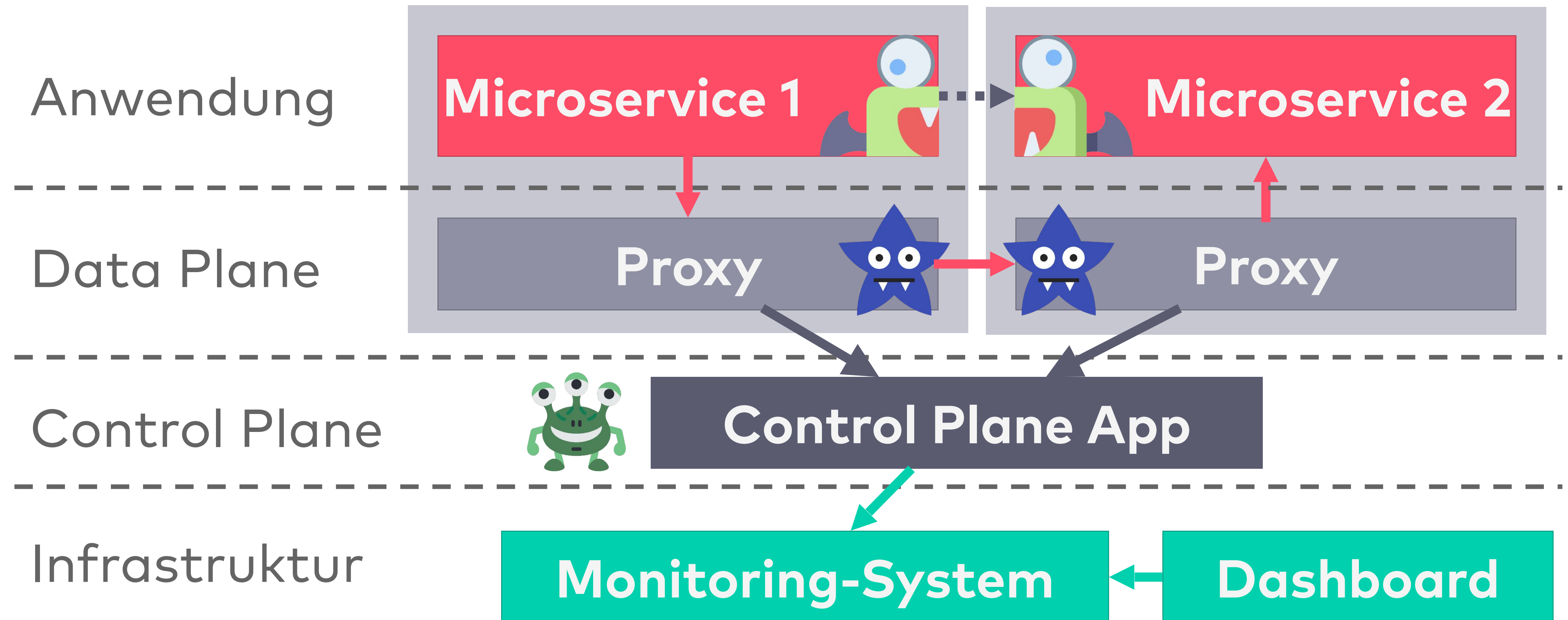
Requests/Sekunde

Status Codes (Fehler)

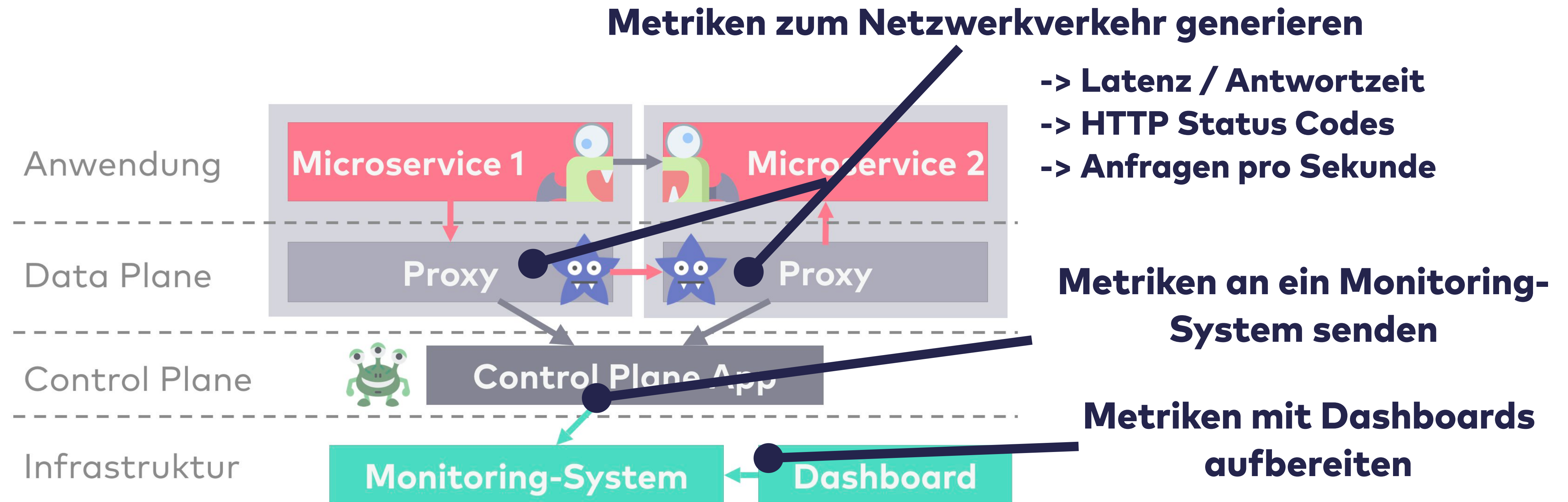
Auslastung

... es hat aber keinen Einblick in die Microservices

Monitoring mit Service Mesh



Monitoring mit Service Mesh





Monitoring mit Istio

Monitoring mit Istio

basiert auf...

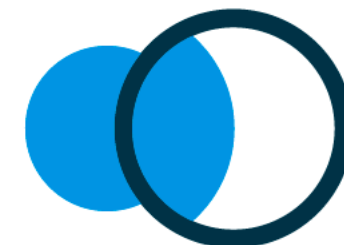
Prometheus



Grafana



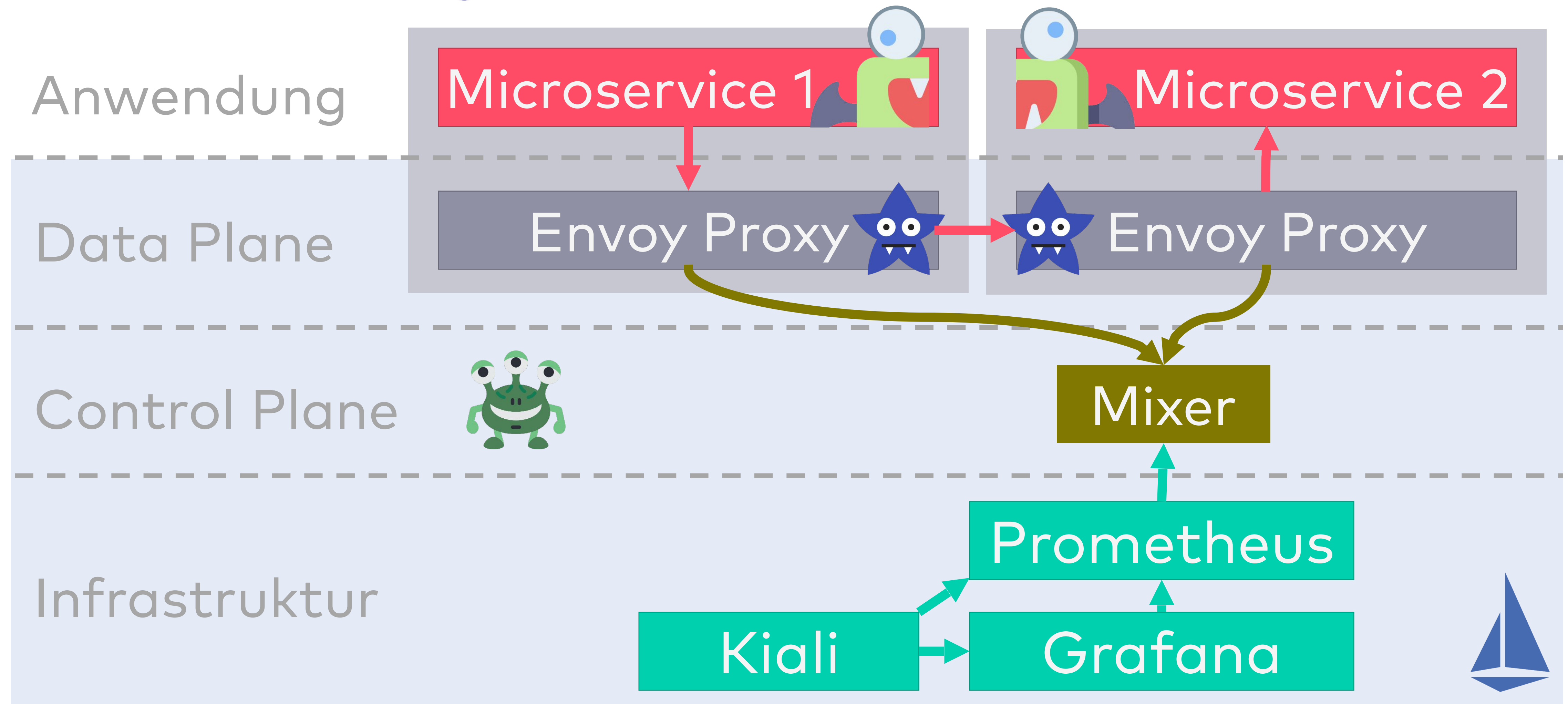
Kiali



Jaeger

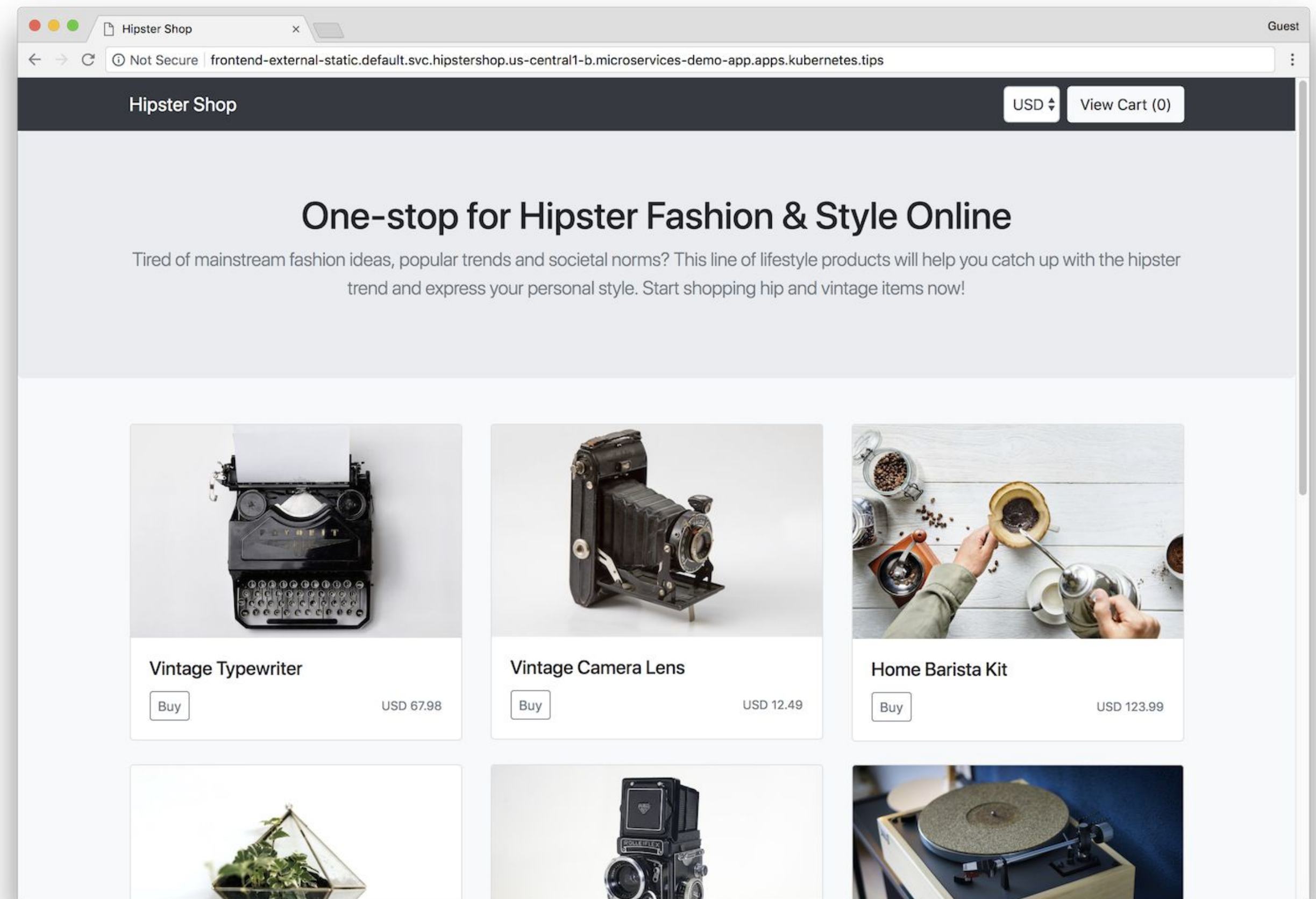


Monitoring mit Istio

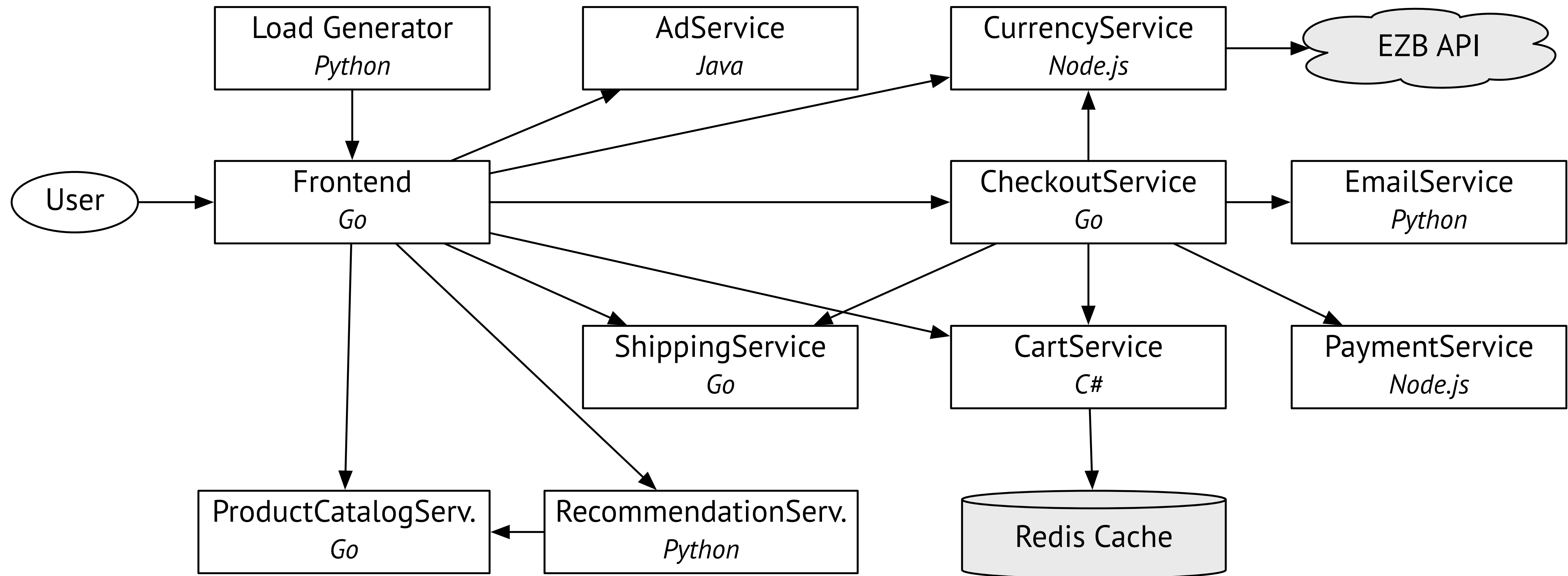


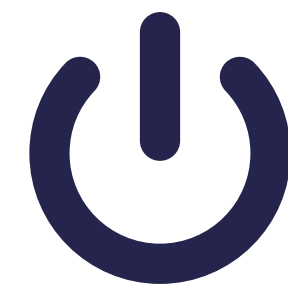
Beispielanwendung: Hipstershop

- Fachlich:
 - Online-Shop für Hipsterbedarf
- Technologisch:
 - **10 Microservices** mit verschiedenen Technologien
 - Aufruf einer **externen API** für Währungsumrechnungen
 - **Redis** für Warenkorb
 - Kommunikation über **gRPC**



Hipstershop Services





Demo

the hipster trend and express your personal style. Start shopping hip and vintage items now!



Buy

USD 67.99



Buy

USD 12.49



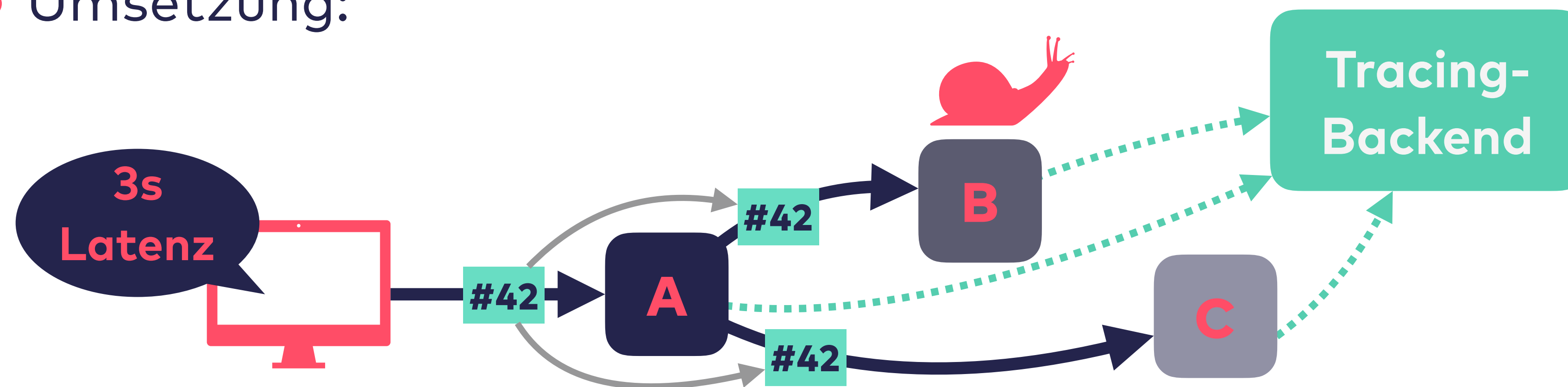
Buy

USD 123.99

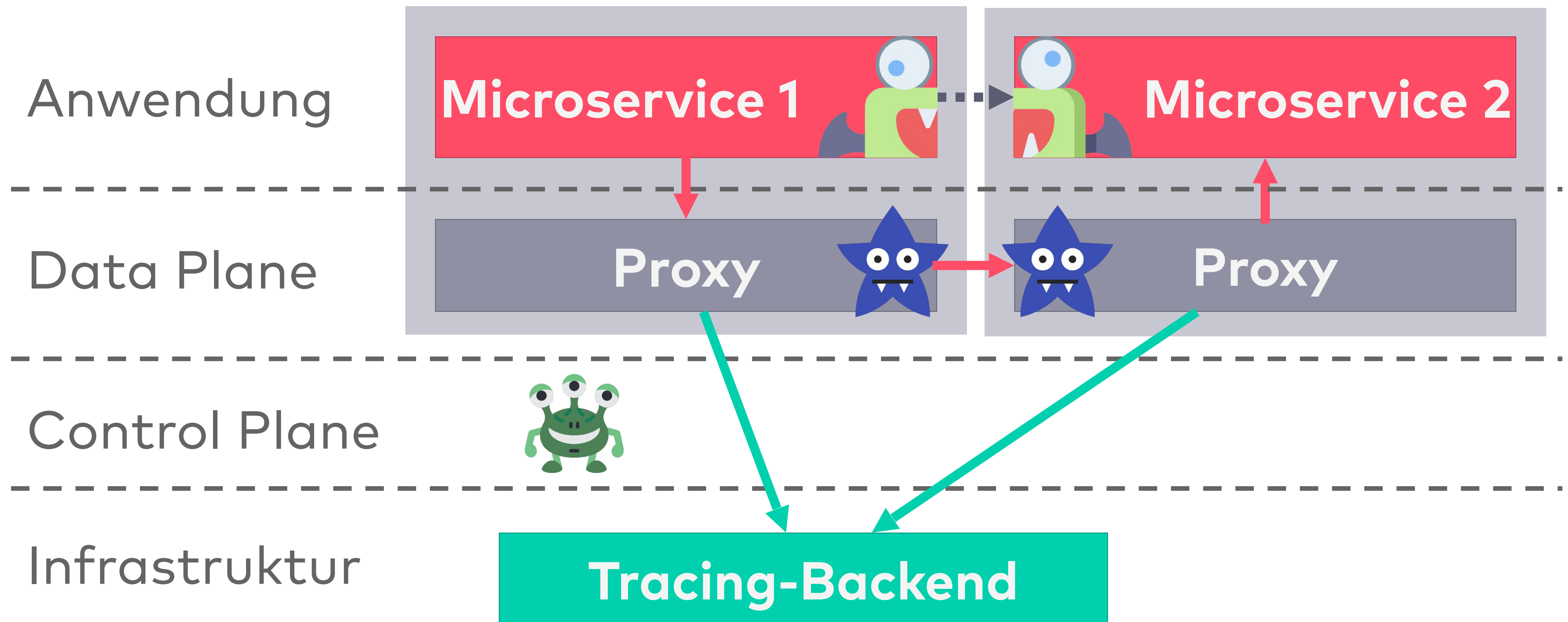


Tracing

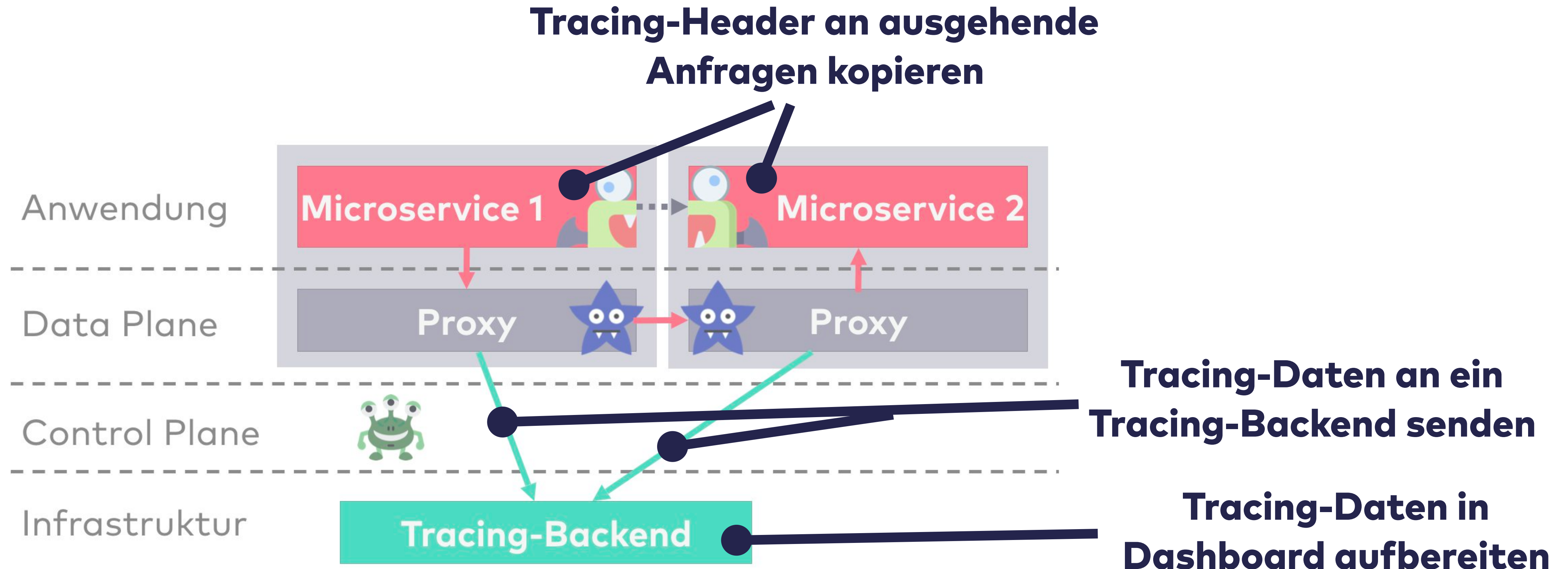
- Beantwortet...
 - Welcher Aufruf erzeugt welchen **Folgeaufruf**?
 - Welchen Anteil hat jeder Teilaufruf an der **Latenz**?
 - Wo ist ein **Fehler** entstanden?
- Umsetzung:

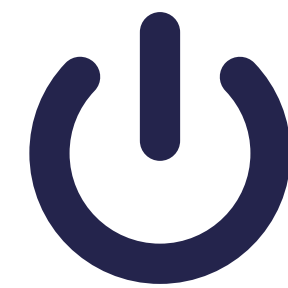


Tracing mit Service Mesh | Config



Tracing mit Service Mesh





Demo

```
istioctl (istioctl)
(* listio:default)→ istio-demo INGRESS_HOST="$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.status.loadBalancer.ingress[0].ip}')"
echo http://"$INGRESS_HOST"
http://34.90.62.137
(* listio:default)→ istio-demo kubectl apply -f istio-config/01_loadgenerator.yaml
pod/loadgenerator created
(* listio:default)→ istio-demo istioctl dashboard kiali
http://localhost:60262/kiali
```

~ (zsh)

Last login: Wed Nov 13 00:27:15 on ttys028

```
(* listio:default)→ ~
```

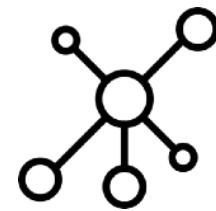
```
demo.sh
1 # Monitoring | Ingress IP
2 INGRESS_HOST="$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.status.loadBalancer.ingress[0].ip}')"
3 echo http://"$INGRESS_HOST"
4
5 # IP-Adresse zeigen
6
7 # -----M-O-N-I-T-O-R-I-N-G-----
8
9 # Load?
10 kubectl apply -f istio-config/01_loadgenerator.yaml
11
12 # Monitoring | Kiali
13 istioctl dashboard kiali
14
15 # Monitoring | Jaeger
16 istioctl dashboard jaeger
17
18 # -----R-O-U-T-I-N-G-----
19
20 # Frontend Version 2
21 open 02_frontend_v2.yaml
22 kubectl apply -f istio-config/02_frontend_v2.yaml
23 kubectl get pods
24
25 # Frontend Version nach URL
26 # Virtual Service
27 open istio-config/03_frontend-virtual-service-uri.yaml
28 kubectl apply -f istio-config/03_frontend-virtual-service-uri.yaml
29 # Destination Rule
30 open istio-config/04_frontend-destination-rule.yaml
31 kubectl apply -f istio-config/04_frontend-destination-rule.yaml
32
33 # Prozentuales Routing
34 open istio-config/05_routing-percent.yaml
35 kubectl apply -f istio-config/05_routing-percent.yaml
36
37 # Test
38 for i in $(seq 20); do curl -s http://$INGRESS_HOST | grep 'Hipster Shop' | grep -v '<title>'; done;
```

@JoergM @HannaPrinz

Service Mesh Features



Observability



Routing

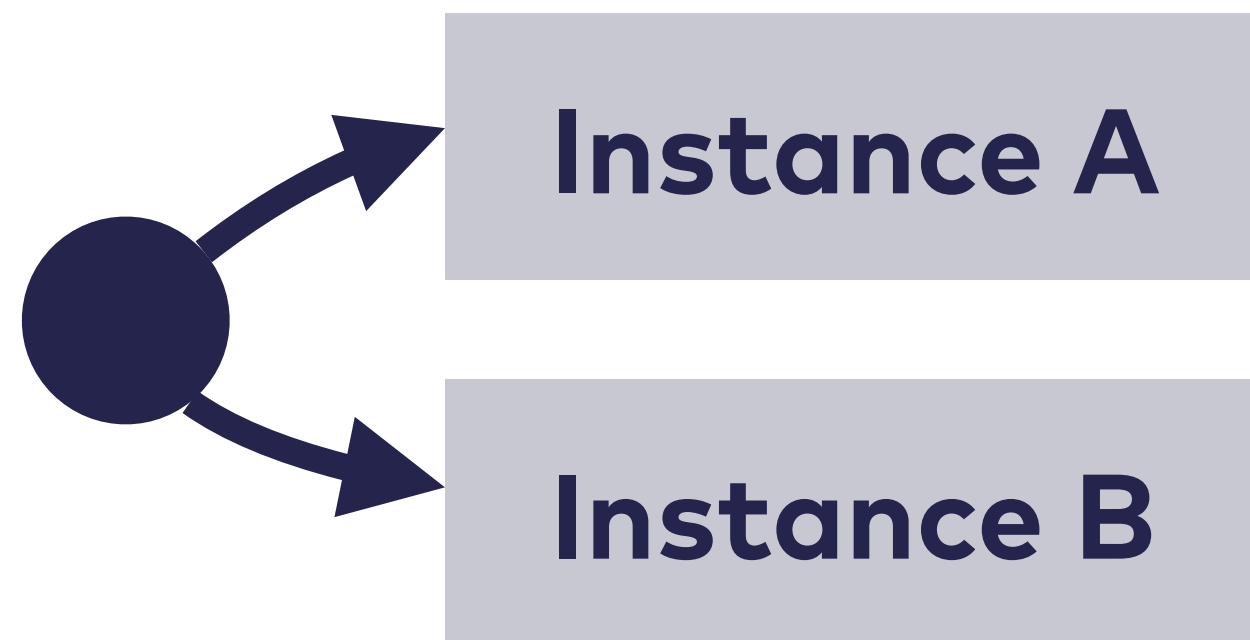


Resilience

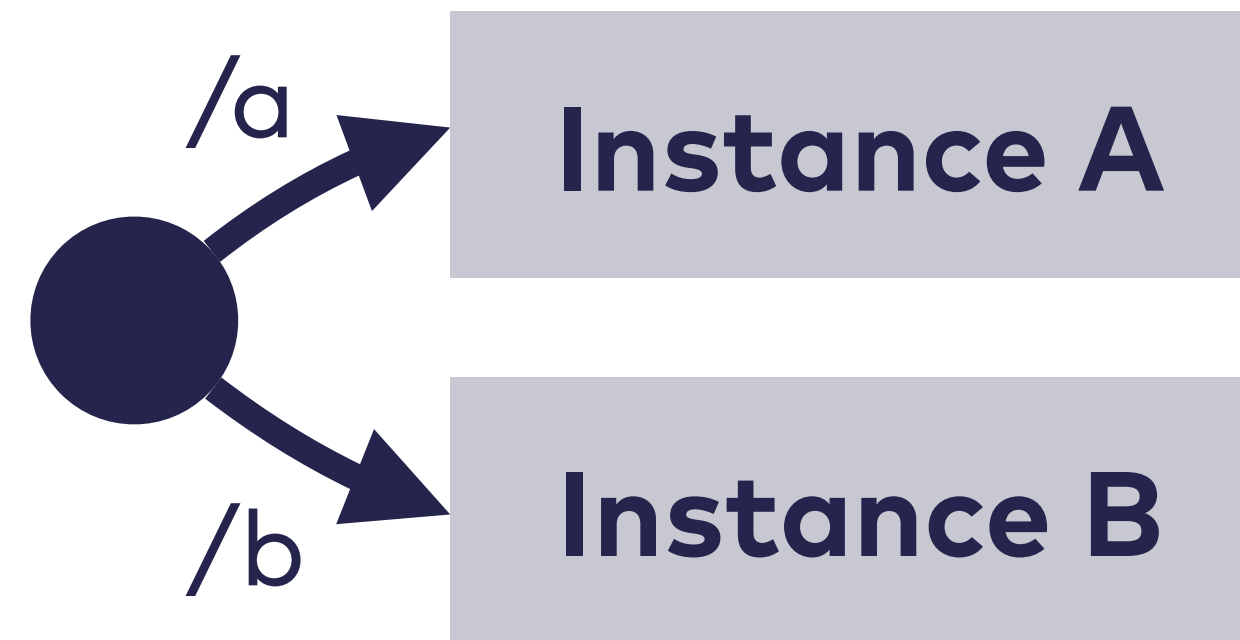


Security

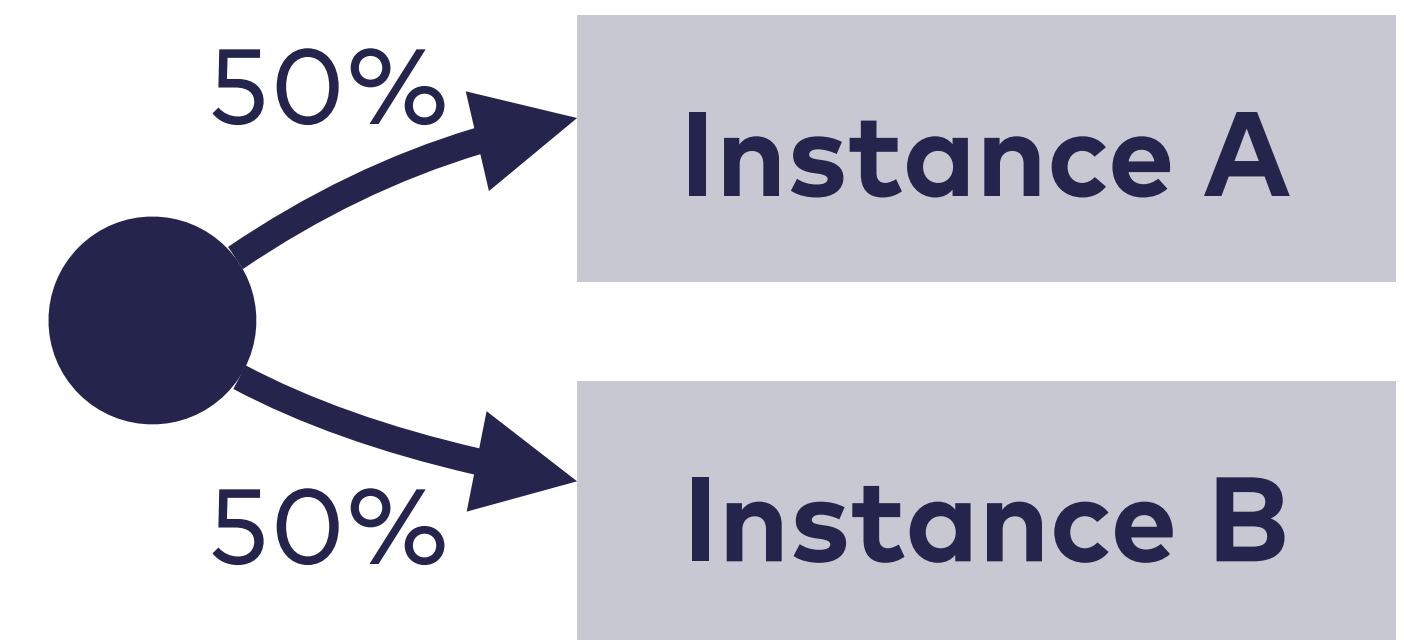
Routing



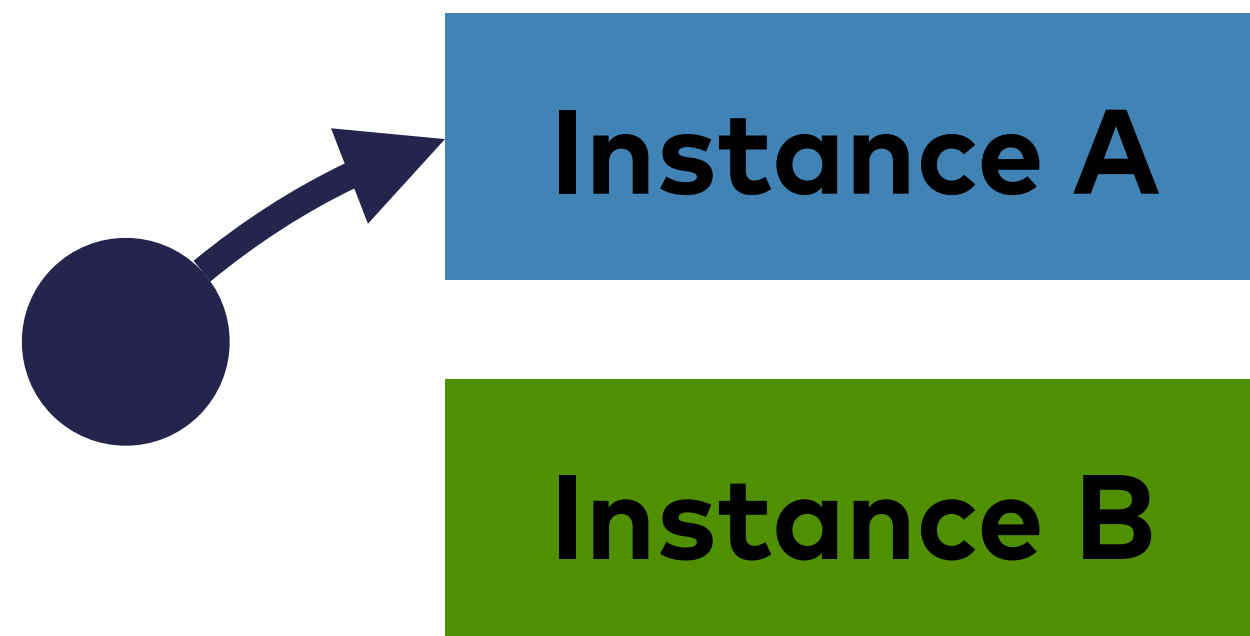
Load Balancing



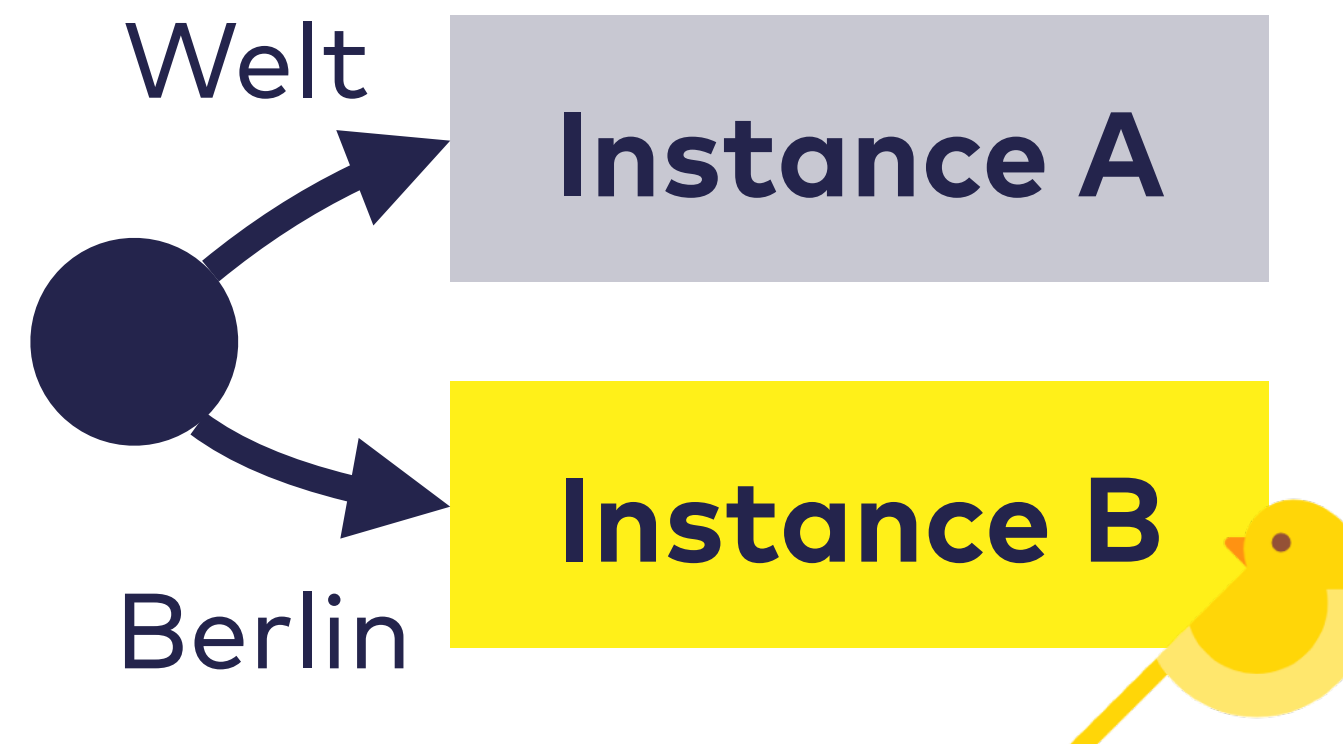
Pfad-basiertes Routing



A/B-Testing



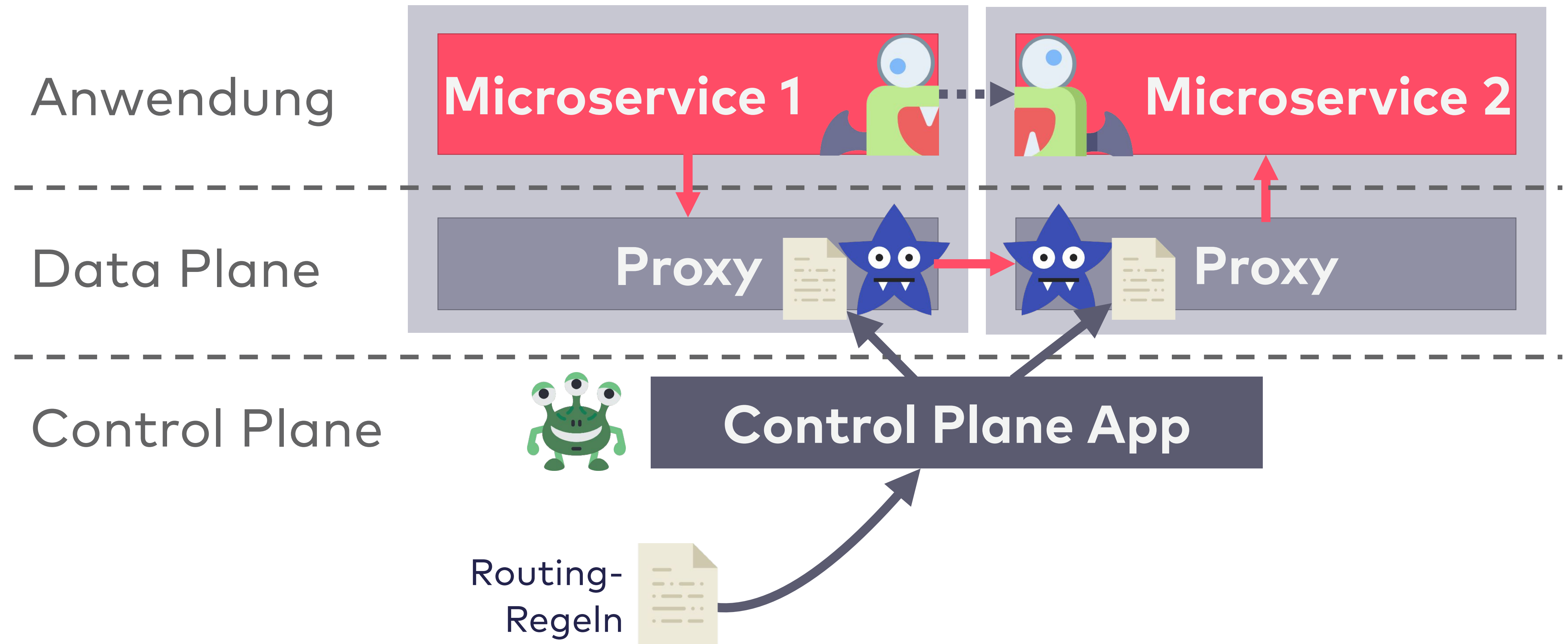
Blue/Green Deployment



Canary Releasing

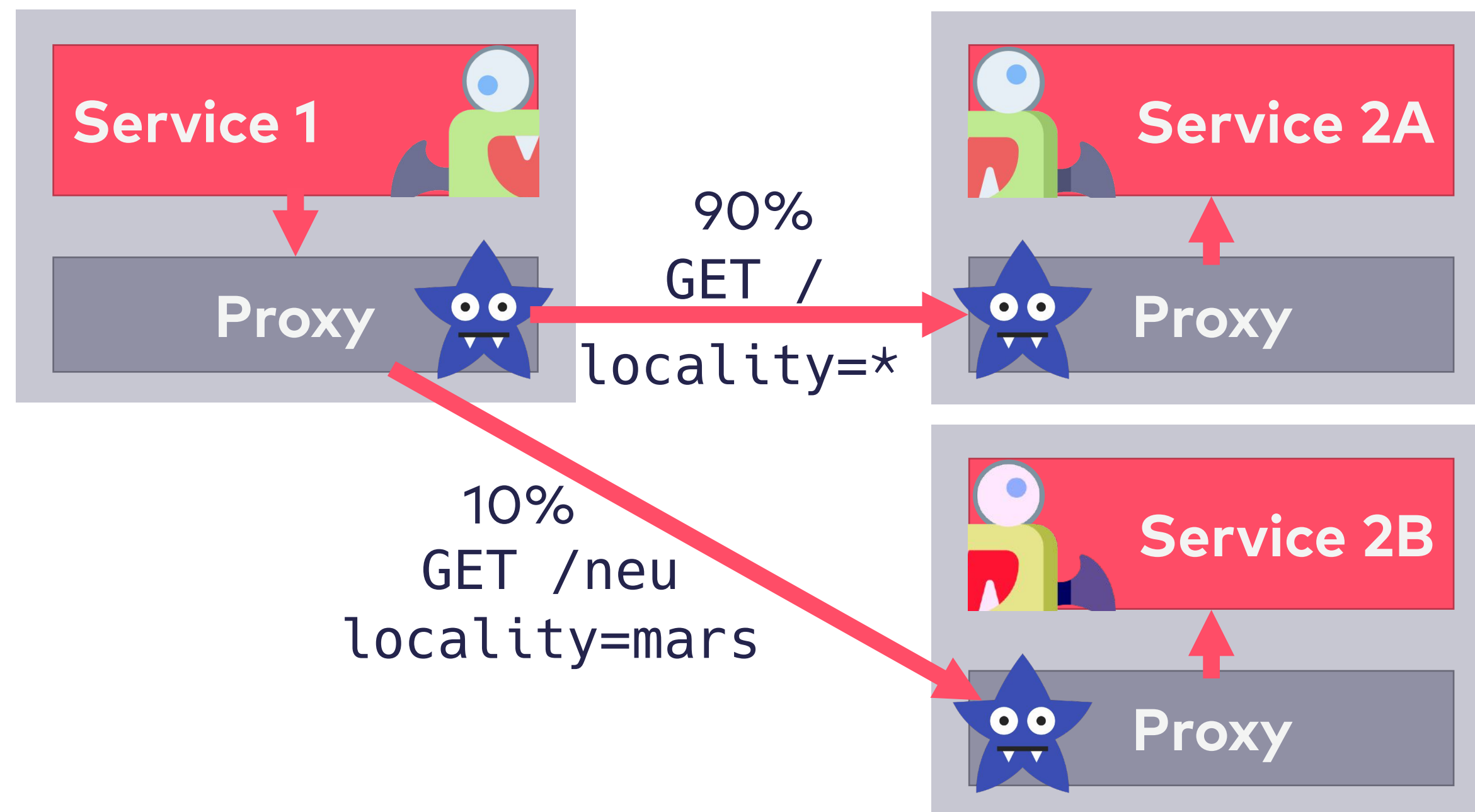
Typischerweise im
**Edge Router / API-
Gateway** implementiert
z.B. NGINX, Kong,
Ambassador, Traefik

Routing mit Service Mesh | Config

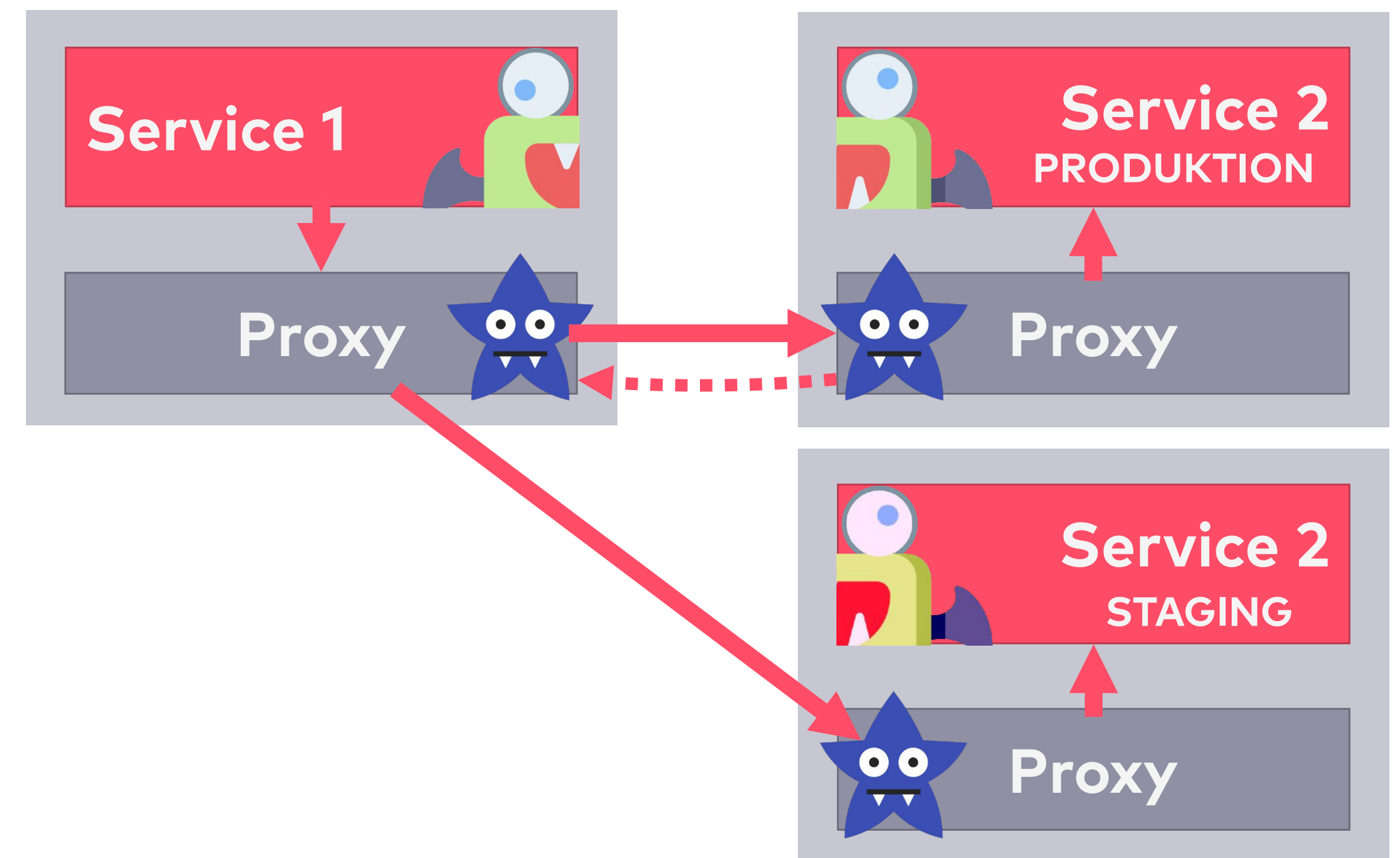


Routing mit Service Mesh

Komplexe Routing-Regeln
für A/B Testing und Canary Releasing



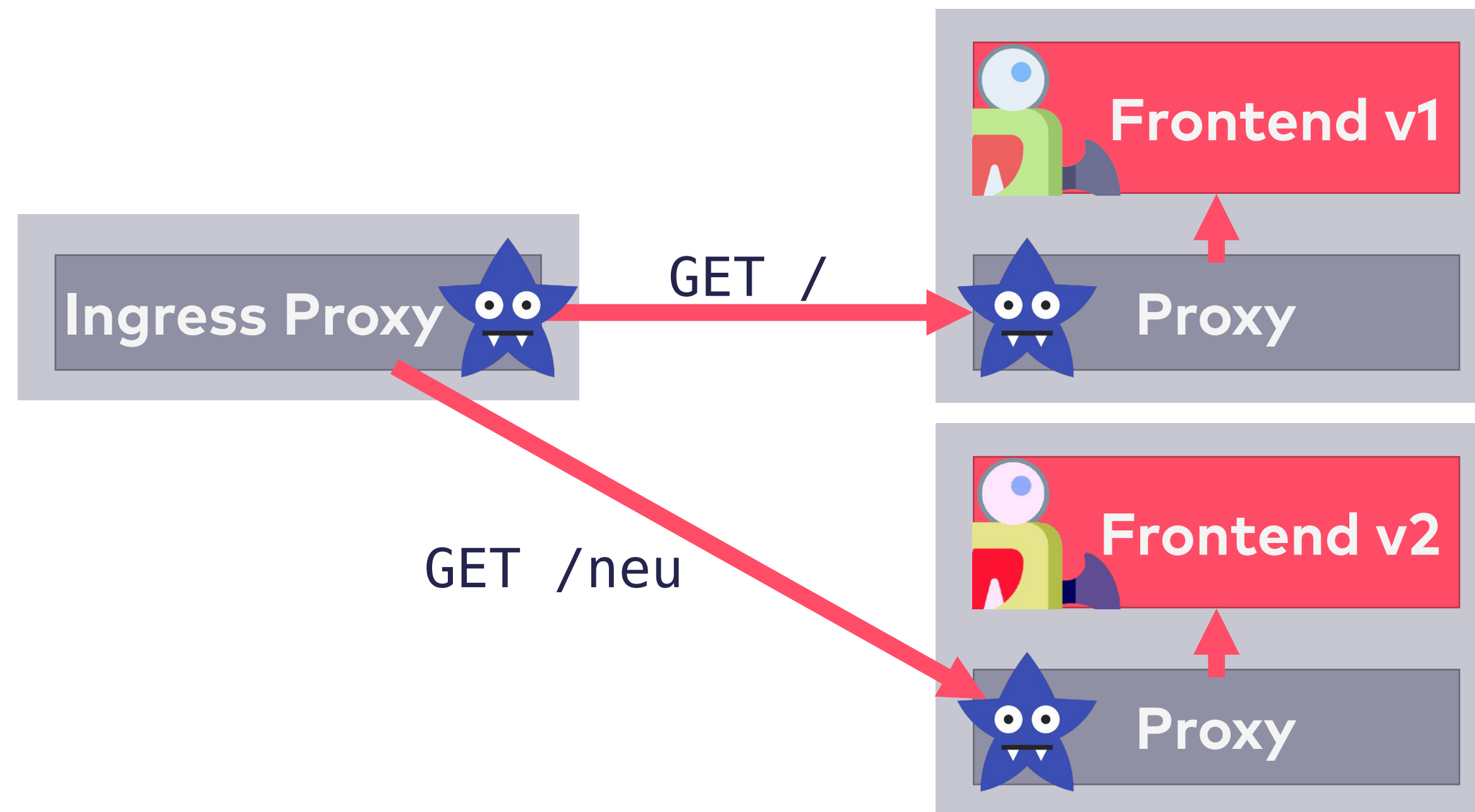
Traffic Mirroring





Routing mit Istio

Beispiel-Routing mit Istio



Routing Config

```
apiVersion: networking.istio.io/...  
kind: DestinationRule  
metadata:  
  name: frontend
```

```
spec:  
  host: frontend  
  subsets:
```

```
- name: v2  
  labels:  
    version: "2"
```

Subset "**v2**":
Alle Pods mit Label
version: "**2**"

```
- name: v1  
  labels:  
    version: "1"
```

Subset "**v1**":
Alle Pods mit Label
version: "**1**"

```
apiVersion: networking.istio.io/...  
kind: VirtualService  
metadata:  
  name: frontend-ingress  
spec:  
  hosts:  
  - "*"   
  http:
```

```
- match:  
  - uri:  
      prefix: "/neu"  
    route:  
      - destination:  
          host: frontend  
            subset: v2
```

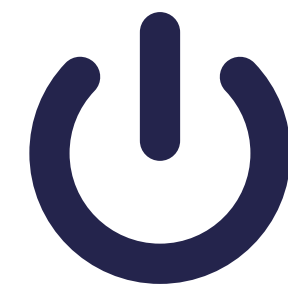
Regel 1:

Wenn die **URI "/neu"**
enthält, dann leite an
frontend **v2** weiter

```
- route:  
  - destination:  
      host: frontend  
        subset: v1
```

Regel 2:

Ansonsten leite an
frontend **v1** weiter



Demo

```
(*) istio:default)→ istio-demo INGRESS_HOST="$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.status.loadBalancer.ingress[0].ip}')"
echo http://"$INGRESS_HOST"
http://34.90.62.137
(*) istio:default)→ istio-demo kubectl apply -f istio-config/01_loadgenerator.yaml
pod/loadgenerator created
(*) istio:default)→ istio-demo istioctl dashboard kiali
http://localhost:60262/kiali
^C%
(*) istio:default)→ istio-demo istioctl dashboard jaeger
http://localhost:60281
^C%
(*) istio:default)→ istio-demo open 02_frontend_v2.yaml
The file /Users/hanna/istio-demo/02_frontend_v2.yaml does not exist.
(*) istio:default)→ istio-demo
```

Last login: Wed Nov 13 00:27:15 on ttys028

```
(*) istio:default)→ ~
```

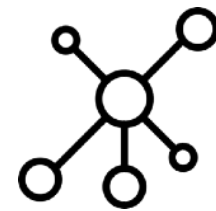
```
1 # Monitoring | Ingress IP
2 INGRESS_HOST="$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.status.loadBalancer.ingress[0].ip}')"
3 echo http://"$INGRESS_HOST"
4
5 # IP-Adresse zeigen
6
7 # -----M-O-N-I-T-O-R-I-N-G-----
8
9 # Load?
10 kubectl apply -f istio-config/01_loadgenerator.yaml
11
12 # Monitoring | Kiali
13 istioctl dashboard kiali
14
15 # Monitoring | Jaeger
16 istioctl dashboard jaeger
17
18 # -----R-O-U-T-I-N-G-----
19
20 # Frontend Version 2
21 open 02_frontend_v2.yaml
22 kubectl apply -f istio-config/02_frontend_v2.yaml
23 kubectl get pods
24
25 # Frontend Version nach URL
26 # Virtual Service
27 open istio-config/03_frontend-virtual-service-uri.yaml
28 kubectl apply -f istio-config/03_frontend-virtual-service-uri.yaml
29 # Destination Rule
30 open istio-config/04_frontend-destination-rule.yaml
31 kubectl apply -f istio-config/04_frontend-destination-rule.yaml
32
33 # Prozentuales Routing
34 open istio-config/05_routing-percent.yaml
35 kubectl apply -f istio-config/05_routing-percent.yaml
36
37 # Test
38 for i in $(seq 20); do curl -s http://$INGRESS_HOST |
```

```
grep 'Hipster Shop' | grep -v '<title>': done;
```

Service Mesh Features



Observability



Routing



Resilience



Security

Resilience

Was wenn ein Service nicht wie erwartet zur Verfügung steht?



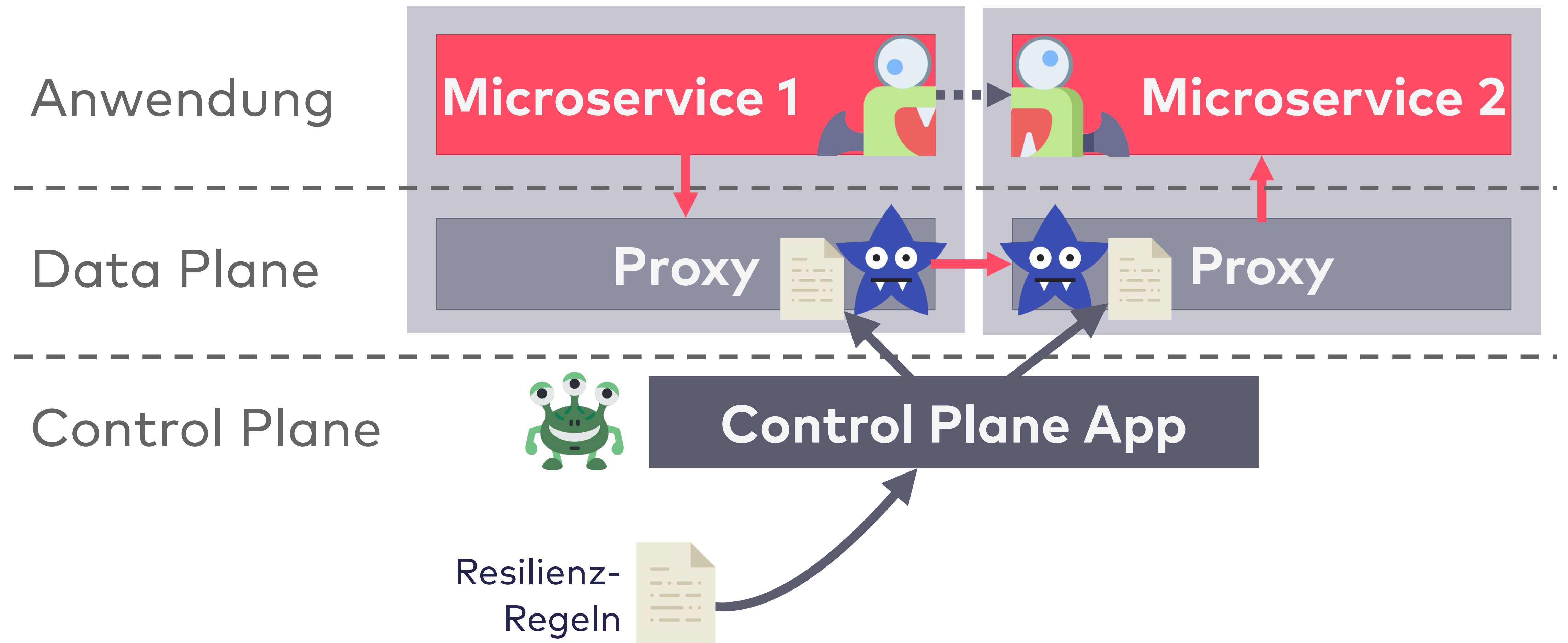
Ziel:

Gesamtsystem funktioniert weiter
... ggf. mit Einschränkungen

Maßnahmen:

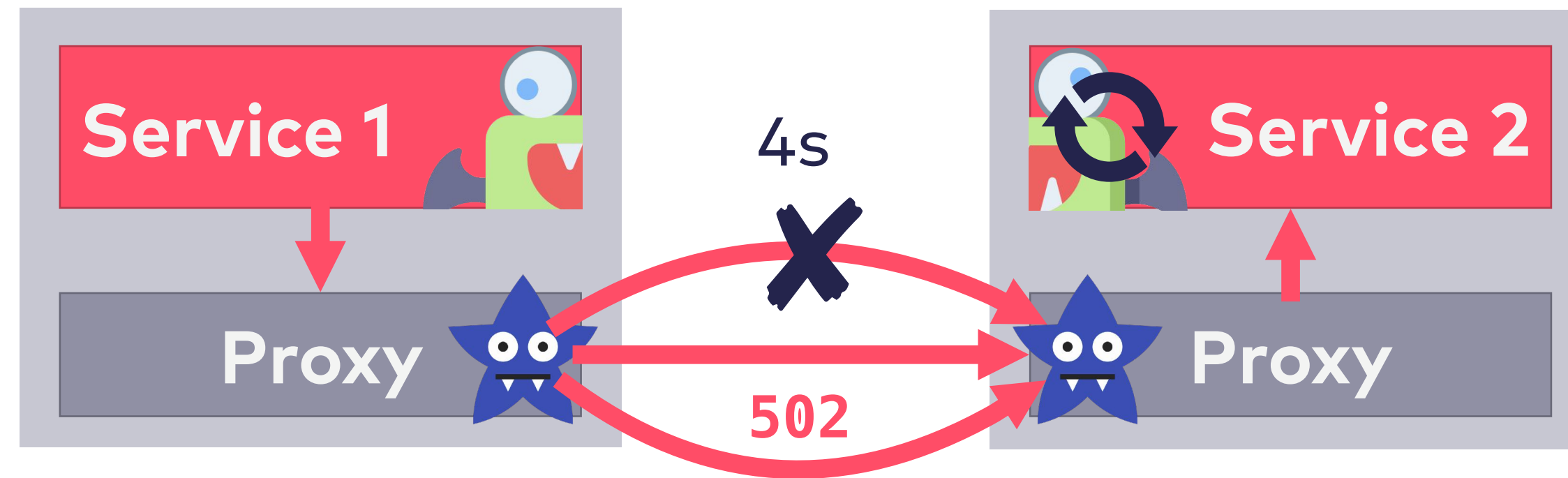
Retry, Timeout, Circuit Breaking

Resilience mit Service Mesh | Config

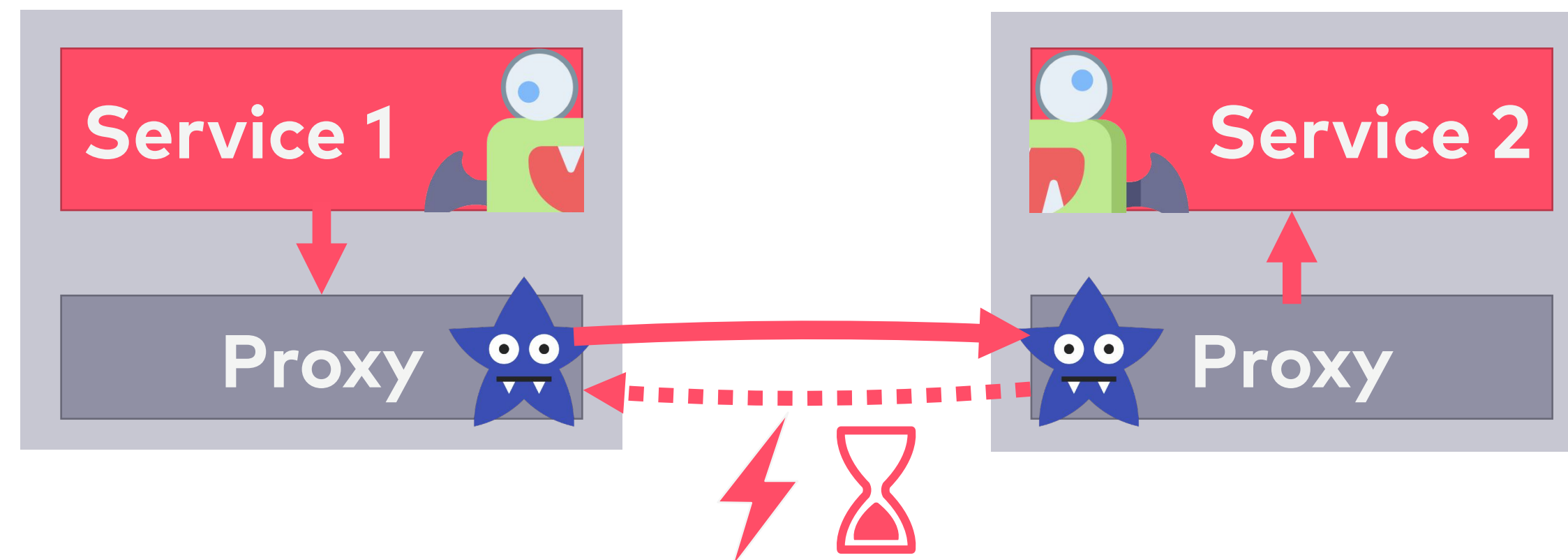


Resilience mit Service Mesh

**Timeout
Retry**



**Fault Injection
Delay Injection**





Resilience mit Istio

Retry & Timeout

Timeout & Retry

hosts: Zieladresse
des Client-Requests

max. Anzahl der Retrys

Timeout pro Retry

Bedingungen für ein Retry

Timeout für alle Retrys zusammen

destination: Wo der Request
tatsächlich hingeht

```
apiVersion: networking.istio.io/...  
kind: VirtualService  
metadata:  
name: recommendationervice  
spec:  
  hosts:  
  - recommendationervice  
  http:  
  - retries:  
    attempts: 3  
    perTryTimeout: 1s  
    retryOn: connect-failure,5xx  
  timeout: 2s  
  route:  
  - destination:  
    host: recommendationervice
```

Service Mesh Features



Observability



Routing

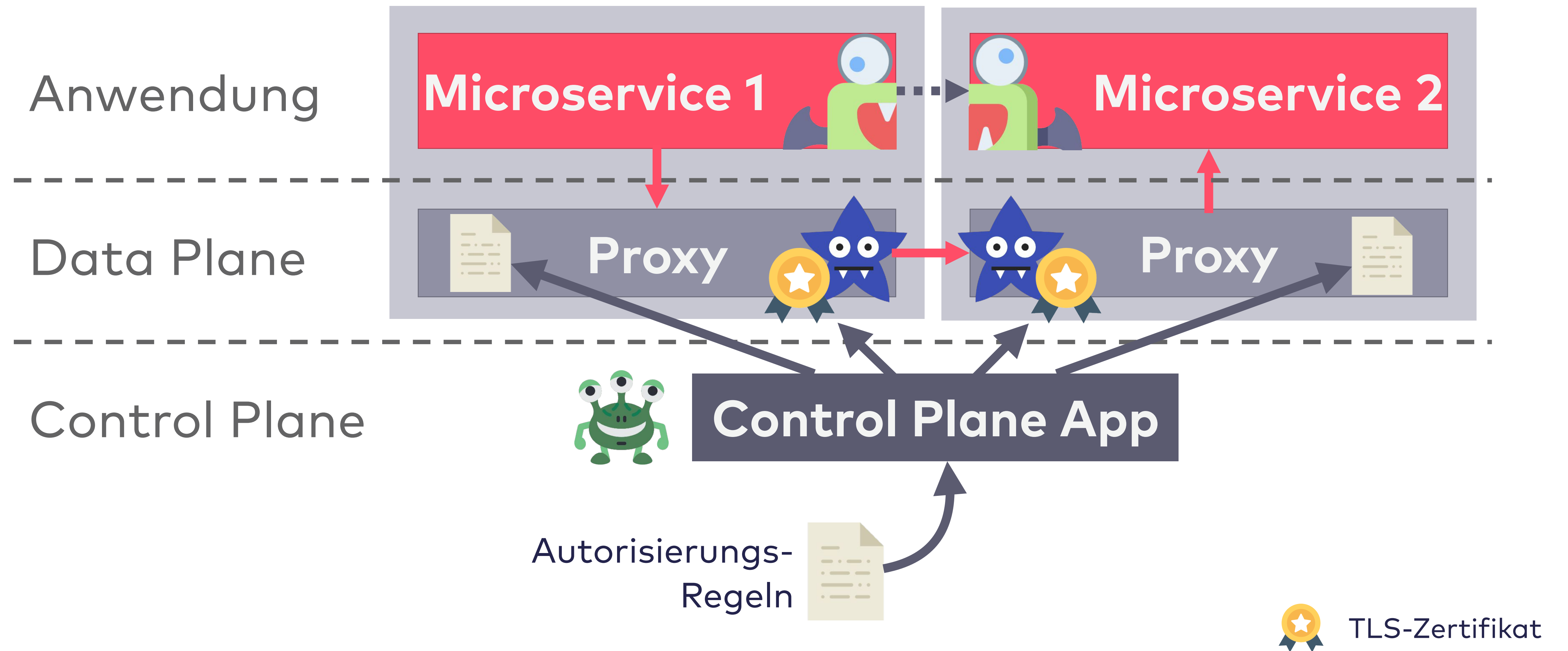


Resilience



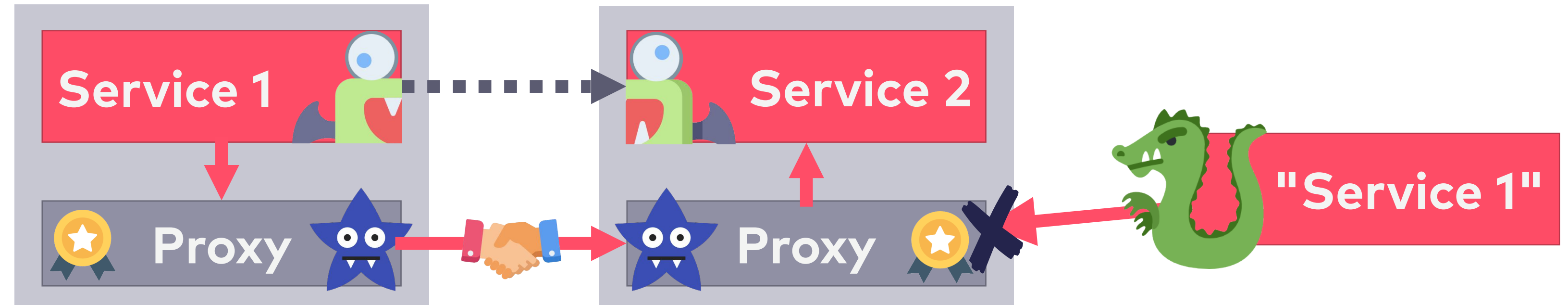
Security

Security mit Service Mesh | Config

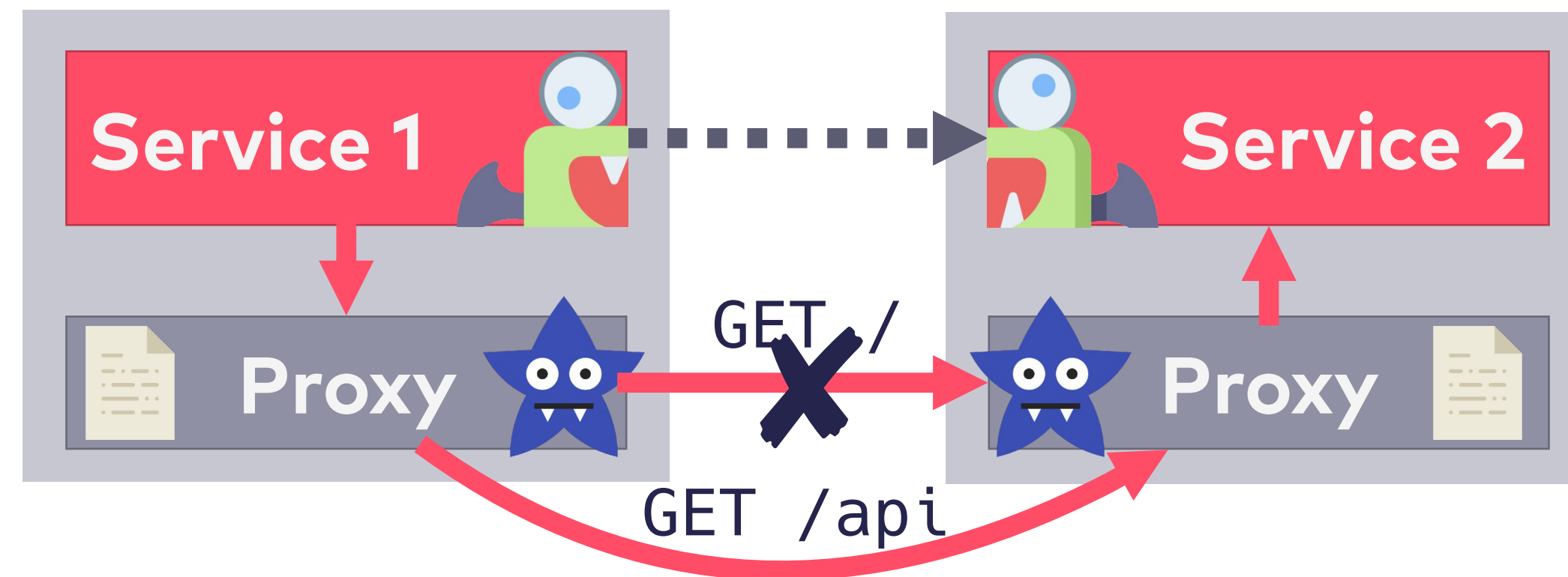


Security mit Service Mesh

Authentifizierung mit mTLS



Autorisierung



TLS-Zertifikat



Autorisierungs-Regel

Service Mesh Features

Netzwerk-Metriken und -Logs
Daten an Tracing-Backend senden



Observability

Business-Metriken oder -Logs
Weitergabe von Tracing-Headern

Komplexe Routing-Regeln
Canary Releasing & A/B-Testing



Routing

Routing anhand von Request Body
Automatisierung Canary Releasing

Automatische Timeouts & Retrys
Automatisches Circuit Breaking



Resilience

Cache oder Standardantworten in
Circuit Breaker nutzen

Authentifizierung mit mTLS
Autorisierung



Security

One Mesh

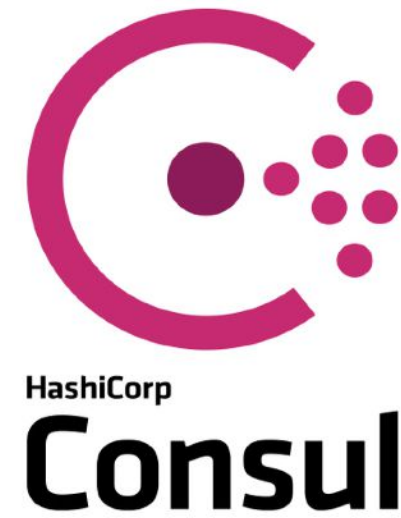


to rule them all?

Service Mesh Implementierungen



LINKERD



AWS App Mesh

mæsh



Istio

Vergleich auf
servicemesh.es



Kuma



	Istio	Linkerd 2	AWS App Mesh	Consul Connect	Maesh	Kuma
Current version	1.4	2.6		1.6	1.0	0.3
License	Apache License 2.0	Apache License 2.0	Closed Source	Mozilla License	Apache License 2.0	Apache License 2.0
Developed by	Google, IBM, Lyft	Buoyant	AWS	HashiCorp	containous	Kong
Service Proxy	Envoy	linkerd-proxy	Envoy	defaults to Envoy , exchangeable	Traefik	Envoy
Ingress Controller	Envoy / Own Concept	any		any	any	any
Governance	see Istio Community	see Linkerd Governance and CNCF Charter	AWS	see Contributing to Consul	see Contributing notice	see Contributing notice
Tutorial	Istio Tasks	Linkerd Tasks	AWS App Mesh Getting Started	Hashicorp Learn platform	Maesh Example	Kuma Kubernetes Quickstart
Platform	Kubernetes, Consul & Nomad	Kubernetes	ECS, Fargate, EKS, EC2	Consul, Nomad, Kubernetes	Kubernetes	Kubernetes, VMs (Universal)
Automatic Sidecar Injection	yes	yes	yes	yes	yes (per Node)	yes
Used in production	yes	yes				
Advantages	Istio can be adapted and extended like no other Mesh. Its many features are available for Kubernetes and other	Linkerd 2 is designed to be non-invasive and is optimized for performance and usability. Therefore, it requires little	AWS App Mesh is integrated into the AWS landscape and	Consul Connect can be used in any Consul environment and therefore does not require a scheduler. The	Maesh focuses on a selection of features to achieve good	Kuma supports both Kubernetes and plain VMs and allows you to

Was ist wichtig für die Auswahl eines Service Mesh?

Features

Kompatibilität

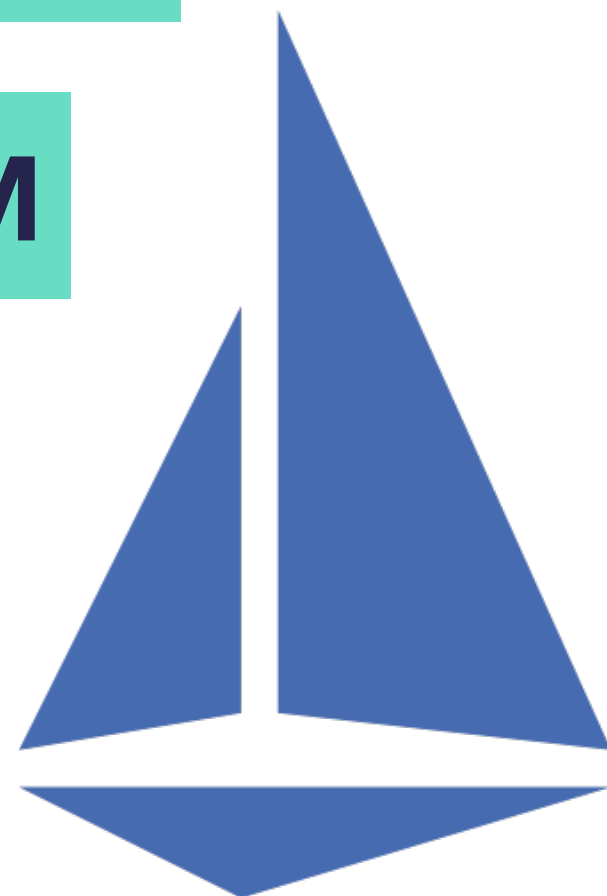
Performance

Usability

***2017**

von **Google & IBM**

optimiert auf
Featureanzahl und
Konfigurierbarkeit



optimiert für
Kubernetes, aber
nicht exklusiv

Istio VS Linkerd 2

***2017**

von **Buoyant**

optimiert auf
Usability und
Performance



Kubernetes only

Features



Istio 1.4



Linkerd 2.6

Generell	Plattform
	Automatische Sidecar-Injektion
Monitoring	Metriken
	Tracing
	Dashboard mit Service Graph
	Load Balancing
Routing	Routing-Regeln
	Timeouts & Retrys
Resilienz	Circuit Breaker
	Authentifizierung via mTLS
Sicherheit	Autorisierung

Verschiedene	Kubernetes
✓	✓
✓	✓
✓	(✓)
✓	✓
✓	✓
✓	✓
✓	✓
✓	✗
✓	✓
✓	✗

LINKERD <

Overview

Tap

Top

Top Routes

Service Mesh

Resources

Documentation

Community

Join the Mailing List

Join us on Slack

File an Issue

Running Linkerd 2.4.0 (stable).

Linkerd is up to date.

deployment/web

meshed

deploy/vote-bot

SR94.02%

RPS1.95

P999 ms

→

deploy/web

SR93.16%

RPS1.95

P995 ms

→

deploy/voting

SR86.21%

RPS0.97

P991 ms

→

deploy/emoji

SR100.00%

RPS1.97

P991 ms

LIVE CALLS

ROUTE METRICS

deployment/web

Route ↑	Service ↑	↑ Success Rate	↑ RPS	↑ P50 Latency	↑ P95 Latency	↑ P99 Latency
GET /api/list	web-svc	100.00% ●	0.98	2 ms	2 ms	3 ms
GET /api/vote	web-svc	86.21% ●	0.97	3 ms	4 ms	5 ms

Performance & Ressourcen

- **Latenz**

- **Istio:** zusätzlich ca. 8ms Latenz - pro Aufruf zwischen Services!
- **Linkerd 2:** ähnlich, mit weniger starken Schwankungen

- **Ressourcen**

- Zusätzliche Container für Control Plane & jedes Sidecar
- → Erhöhter CPU- & Arbeitsspeicher-Verbrauch

**Aber: Abhängig von konkretem Setting
→ eigenen Benchmark machen!**

Usability



Dave Strebel
@dave_strebel

"Finally got Istio into production"



2:21 AM · Sep 18, 2019 from [Apple Valley, MN](#) · [Tweetbot for iOS](#)

322 Retweets **1.2K** Likes

Service Mesh Fazit



**Löst viele wesentliche Probleme
von Microservices**

... ohne Codeänderungen!



**Eine weitere komplexe
Technologie**

**Latenz- und Ressourcen-
Overhead**

Entscheidungshilfe

Mesh oder kein Mesh?

- **Service Mesh** wenn:
 - Großteil in Kubernetes läuft
 - Viele Microservices, viele synchrone Aufrufe
 - Viele ungelöste Probleme beim Monitoring, Routing, Resilienz und/oder Security

Istio oder Linkerd 2?

- **Istio nur** wenn Komplexität und Flexibilität nötig ist:
 - Teilsystem bleibt außerhalb von Kubernetes
 - Circuit Breaking und Autorisierung unverzichtbar
 - Ausreichend zeitliche und kognitive Kapazität im Team

Mehr Mesh

- Artikel **Brauchen asynchrone Microservices und Self-Contained-Systems ein Service Mesh?**
- Artikel **Alle 11 Minuten verliebt sich ein Microservice in Linkerd**
- **Service Mesh Vergleich auf servicemesh.es**
- Podcast zu Service Meshes
- Beispiel-Anwendung auf GitHub
- Tutorial von Linkerd
- Tutorial von Istio

Danke! Fragen?

Jörg Müller
joerg.mueller@innoq.com
@JoergM

Hanna Prinz
hanna.prinz@innoq.com
@HannaPrinz



INNOQ
www.innoq.com

Icons made by [srip](#),
[Smashicons](#), [Nikita Golubev](#), [Freepik](#), [surang](#)
and [Darius Dan](#) from
www.flaticon.com and
licensed by [CC 3.0 BY](#)

Kostenloser Service Mesh Primer
Auf leanpub.com/service-mesh-primer

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim am Rhein
Germany
+49 2173 3366-0

Ohlauer Str. 43
10999 Berlin
Germany
+49 2173 3366-0

Ludwigstr. 180E
63067 Offenbach
Germany
+49 2173 3366-0

Kreuzstr. 16
80331 München
Germany
+49 2173 3366-0

Hermannstrasse 13
20095 Hamburg
Germany
+49 2173 3366-0

innoQ Schweiz GmbH

Gewerbestr. 11
CH-6330 Cham
Switzerland
+41 41 743 0116