OBJEKTspektrum

Service Meshes in Microservice Architekturen

Transparente Sicherheit, Beobachtbarkeit und Zuverlässigkeit für Cloud-native Anwendungen

TECHNISCHE HOCHSCHULE LÜBECK

CoSA Center of Excellence

Inhaltliche Entwicklung: Prof. Dr. Nane Kratzke, Technische Hochschule Lübeck, Mönkhofer Weg 239, 23562 Lübeck
Inhaltliche Mitwirkung: Dr. Josef Adersberger, QAware GmbH

DEFINITION

Service Meshes erzeugen eine Netzwerkabstraktion für containerbasierte Applikationen und Dienste, um deren Management zu vereinfachen und Aspekte wie Traffic Management, Zuverlässigkeit, Beobachtbarkeit und Sicherheit innerhalb von Microservice-basierten Architekturen zu vereinfachen.

WOZU EIGENTLICH SERVICE MESHES?

Die Entwicklung und Betrieb eines einzelnen Microservice ist zwar einfach (→ GAINS), der Betrieb komplexer Service-of-Service Systeme bringt aber im Gegenzug diverse weitere Herausforderungen mit sich (→ PAINS). Service Meshes lösen diese Herausforderungen außerhalb einer Applikation in einer explizit dafür vorgesehenen Schicht.

EXKURS: SIDECAR

Anwendungen und Dienste benötigen häufig zusammengehörige Funktionen, wie z. B. Überwachung, Protokollierung, Konfiguration, etc.. Solche kohärenten Peripherieaufgaben können als separate Komponenten oder Dienste implementiert werden. Werden diese Aufgaben in einem eigenen Prozess oder Container bereitgestellt, um Plattformdienste sprachübergreifend bereitzustellen, spricht man von einem Sidecar.

Proxies in Service Meshes werden häufig als Sidecars in Containern bereitgestellt, die jeder Serviceinstanz zur Seite gestellt werden.

AINS OF MICROSERVICES

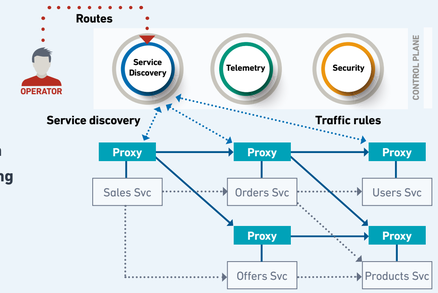
- Einfache Entwicklung (einzelner) Microservices
- Bounded Contexts und Domain Driven Design sind näher an der Denkweise der Fachabteilung
- Cloud-native und DevOps by design
- Unabhängiges Deployment (kein Downtime durch monolithische Rollouts)
- Einfache Integration in Continuous Integration und Continuous Deployment Pipelines

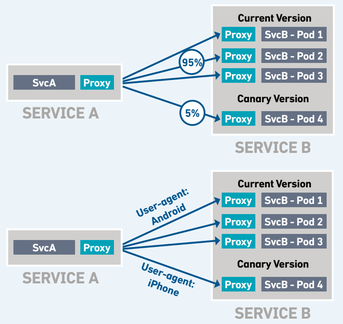
PAINS OF MICROSERVICES

- Inhärente Komplexität
- Geschäftslogik ist über voneinander unabhängige Microservices verteilt
- Sicherheit (Zugriffskontrolle und vergrößerte Attack Surfaces)
- Verteiltes Logging, Traceability

TRAFFIC-MANAGEMENT UND ZUVERLÄSSIGKEIT

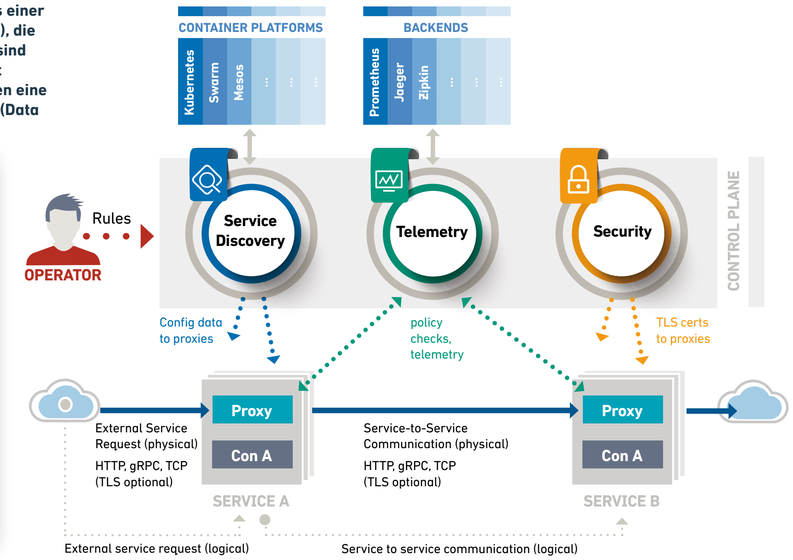
- Last- und zustandsadaptives Service Discovery
- Mit einem selbstheilendem System Design
- Ermöglicht intelligentes (adaptives) Routing
- A/B tests, Canary Deployments, schrittweises Rollout, etc.
- Traffic-Splitting/Steering





Architektur eines Service Meshes

Ein Service Mesh besteht aus einer Kontrollebene (Control plane), die Proxies steuert. Die Proxies sind Serviceinstanzen zugeordnet (Sidecar Pattern) und spannen eine verschlüsselte Datenschicht (Data plane) für Services auf.



PROXY

- Jeder Dienst kommuniziert über einen dedizierten Proxy (Sidecar Pattern).
- Die gesamte Kommunikation im Mesh erfolgt über diese Proxies.
- Proxies können so das Kommunikationsverhalten anreichern und z.B. Tracing- und Monitoring-Daten sammeln.
- Proxies lassen sich bequem von Orchestrierungs-Frameworks verwalten, um so z.B. bei jedem Deployment automatisch platziert zu werden.

CONTROL PLANE

- Verwaltung von Routen und Regeln
- Konfiguration von Timeouts, Retries, etc.
- Service Discovery
- Access Control
- Policy Control
- Telemetriedaten

DATA PLANE

- Eingehende (ingress) und ausgehende (egress) Kommunikation
- Service-to-Service Kommunikation
- Kommunikation über Proxies
- Transparent für verwaltete Dienste

ZUVERLÄSSIGKEIT DURCH RESILIENZ-PATTERNS

- Circuit-Breaker
- Timeouts + Retries
- Recovery
- Latenzbasiertes Load Balancing
- Health-aware Service Discovery

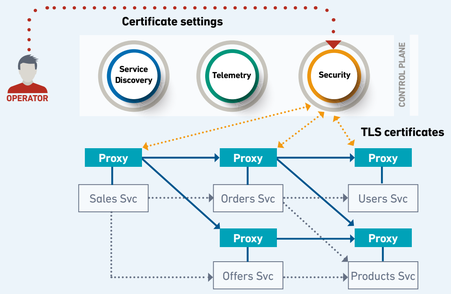
TRAFFIC MANAGEMENT

- Dynamisches regelbasiertes Routing
- Regel- und volumenbasiertes Traffic Splitting
- Rate Limit
- Quotas

SICHERHEIT

Service Meshes folgen häufig einem Secure-by-Default Ansatz und realisieren so eine für Services transparente Verschlüsselung des Datenverkehrs. Ferner werden Kommunikationsregeln flexibler und auf Basis von Serviceidentitäten (logisch) anstatt mittels Network Controls (physisch) durchgesetzt.

- Authentifizierung (Service-to-Service und End-User)
- Authorisation (Service-to-Service und End-User)
- Identitäts- und Credential-Management
- Transparante Verschlüsselung der Service Communication
- Transparentes Zertifikat Handling (TLS)
- TLS Endpunkt Termination



BEISPIELE FÜR SERVICE MESHES UND PROXIES

- Linkerd
- Istio
- Consul
- Kong
- Traefik
- Envoy
- NGINmesh
- ...

GÄNGIGE FEATURES VON SERVICE MESHES

Traffic Management	Resiliency	Security	Observability
• Request Routing	• Timeouts	• mTLS	• Metrics
• Load Balancing	• Circuit Breaker	• Role-Based Access Control	• Logs
• Traffic Shifting	• Health Checks	• Workload Identity	• Traces
• Traffic Mirroring	• Retries	• Authentication Policies	
• Service Discovery	• Rate Limiting	• CORS Handling	
• Ingress, Egress	• Delay & Fault Injection	• TLS Termination	
• API Specification	• Connection Pooling		
• Multicluster Mesh			

TELEMETRIE UND BEOBACHTBARKEIT

Telemetriedaten werden transparent durch die Proxies des Service Meshes gesammelt, um Latenz-, Zustands und Volumen-basiertes Load Balancing und Traffic-Management zu ermöglichen. Dies ermöglicht Resilienz und Beobachtbarkeit.

- Tracing
- Monitoring
- Logging
- Erfassung von Telemetriedaten