OMNI52 Service Mesh (tool-agnostisch)

2026.06

Tools im Vergleich

Die drei Patterns

Sidecar (Istio classic, Linkerd): Envoy/linkerd2-proxy neben jedem App-Container. Mature, transparent, Memory-Kosten.

Sidecar-less / eBPF (Cilium): L3/L4 im Kernel, Identity per pod label, kein per-Pod-Datenpfad. L7 via Envoy als shared proxy (`cilium-envoy` DaemonSet).

Ambient (Istio 1.30+ GA): ztunnel als Node-Agent (L4/mTLS), Waypoint-Proxy on-demand (L7). Sidecarless ohne Kernel-Pfad.

Auswahl: was wann

Istio (Sidecar): groesstes Ecosystem, alle L7-Features, Multi-Cluster mature, hohe Memory-Last (50–150 MiB/Pod).
Istio (Ambient): deutlich weniger Overhead, weniger Migration-Reibung, Multi-Cluster jung.
Linkerd: schlank, Rust-Proxy, geringere CPU, weniger L7-Features (kein Wasm), kein Native Multi-Cluster ohne SMI.
Cilium Service Mesh: wenn CNI eh Cilium ist. Identity am Kernel, eBPF-Observability. L7-Features via Envoy-Sidecar (kein Sidecar-less L7).

Datapath / Hop-Latenz

Sidecar (Istio/Linkerd): ~0,5–2 ms p99 pro Hop. Linkerd-Proxy ist 2–3× leichter als Envoy bei einfachem Routing.
Cilium L4: kernel-bypass-frei, Mikrosekunden-Range.
Ambient L4 (ztunnel): unter Sidecar bei reinem mTLS-Forwarding; Waypoint-Hop nur wenn L7-Policy greift.

Traffic Management

Routing-APIs im Vergleich

Istio: `VirtualService` + `DestinationRule` (eigene CRDs) – maechtig, aber zwei Konzepte: Match/Route vs. Subset/LB/CB.
Linkerd: `HTTPRoute` (Gateway API native), `ServiceProfile` fuer Per-Route-Settings.
Cilium: `CiliumNetworkPolicy` (L3-L7) + `HTTPRoute` (Gateway API) – Routing und Policy in einer CRD-Familie.
Konvergenz-Punkt: Gateway API v1.x. Alle drei implementieren sie. Reines Routing → Gateway API, proprietaere CRDs nur fuer Tool-Spezifika (z.B. Wasm).

```yaml
# Gateway API HTTPRoute (alle drei verstehen das)
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata: {name: api}
spec:
  parentRefs: [{name: web-gw, namespace: ingress}]
  hostnames: ["api.example.com"]
  rules:
  - matches: [{path: {type: PathPrefix, value: /v1}}]
    backendRefs: [{name: api, port: 8080}]
```

Traffic-Split / Canary

Istio: `VirtualService.http[].route[].weight`.
Linkerd: `HTTPRoute backendRefs[].weight`.
Cilium: `HTTPRoute` mit weighted backends, alternativ mit Flagger/Argo-Rollouts auf allen drei.

Egress & Circuit-Breaking

Istio: `ServiceEntry` fuer externes DNS-Routing, `DestinationRule.trafficPolicy.outlierDetection` + Connection-Pool.
Linkerd: keine ServiceEntry-Aequivalent, kein eingebauter CB – Retries + Timeouts in ServiceProfile, Resilience-Pattern in der App.
Cilium: FQDN-Policy fuer Egress, keine native Outlier-Detection.

Security & Identity

mTLS & Identity

Istio: SPIFFE-Identity (`spiffe://td/ns/X/sa/Y`), istiod als CA, automatische Cert-Rotation. `PeerAuthentication` STRICT pro Namespace.
Linkerd: identity-Service als CA, Identity per ServiceAccount, mTLS by default ohne Opt-in (sobald beide Pods im Mesh sind).
Cilium: SPIFFE-Identity (1.14+), mTLS optional pro `CiliumNetworkPolicy`. Pod-Identity per Labels, nicht ServiceAccount.

Authorization-Policy

Istio: `AuthorizationPolicy` mit `action: ALLOW|DENY|AUDIT|CUSTOM`. JWT-Claims via `when` oder `RequestAuthentication`.
Linkerd: `ServerAuthorization` (per Server), `HTTPRoute` + `AuthorizationPolicy`. Schlanker, weniger Match-Optionen.
Cilium: L7-Rules in `CiliumNetworkPolicy` (HTTP-Method, Path, Header). JWT nicht nativ.

```yaml
# Istio: default-deny + selektive Erlaubnis
apiVersion: security.istio.io/v1
kind: AuthorizationPolicy
metadata: {name: deny-all, namespace: prod}
spec: {}                          # leere Rules = deny
---
apiVersion: security.istio.io/v1
kind: AuthorizationPolicy
metadata: {name: allow-api, namespace: prod}
spec:
  selector: {matchLabels: {app: api}}
  rules:
  - from: [{source: {principals:
      ["cluster.local/ns/web/sa/frontend"]}}]
```

Observability

Was du out-of-the-box bekommst

Istio: `Telemetry`-API (Metriken, Tracing, Logs) mit Provider-Konfiguration. Prometheus + Kiali + Jaeger als Standard-Stack.
Linkerd: eingebautes Dashboard, Tap (live-traffic), Prometheus-Federation moeglich. Tracing extern via OpenCensus.
Cilium: `Hubble` (eBPF-basiertes Flow-Logging), Hubble UI fuer Service-Map. Metriken in Prometheus, Tracing extern.

Pflicht-Metriken

Golden Signals: Requests/s, Errors/s, Latency-p50/p95/p99 (`istio_requests_total`, `request_duration_milliseconds`, linkerd: `response_latency_ms_bucket`).
Saturation: Sidecar-CPU/Memory, xDS-Push-Latenz (Istio: `pilot_xds_push_time`), Hubble: `hubble_drop_total`.

Tracing-Falle

Alle drei brauchen App-Header-Propagation (`b3`, `traceparent`). Ohne das zerreisst der Span-Tree beim ersten App-zu-App-Hop. Kein Mesh kann das auf L7 simulieren.

Multi-Cluster

Topologien

Istio: Primary-Remote (ein istiod, Workload-Cluster ohne Control-Plane) | Multi-Primary (istiod pro Cluster, gemeinsame Root-CA) | External Control-Plane. Pflicht: gemeinsame `trustDomain` oder `trustDomainAliases`.
Linkerd: `multicluster` extension, Gateway-basiert (Service-Mirror), eine TrustAnchor.
Cilium: `ClusterMesh`, eBPF-direct-routing zwischen Clustern (kein Gateway-Hop) bei Cilium-CNI auf allen Clustern.

Trust-Domain-Falle

Verschiedene Trust-Domains ⇒ mTLS-Reject zwischen Clustern. Migration: `trustDomainAliases` im Quell-Cluster setzen, bevor neue Workloads deployt werden, dann switchen.

Performance & Tuning

Sidecar-Sizing (Faustregel)

Envoy (Istio): 50–150 MiB pro Sidecar bei mittlerer Mesh-Groesse, abhaengig von `Sidecar`-Scope. Pflicht: `Sidecar`-Resource pro Namespace, sonst bekommt jeder Proxy jede Service-Config (Push-Storm bei >500 Services).
linkerd2-proxy (Rust): 10–30 MiB.
ztunnel (Ambient): ~1 ztunnel pro Node, deutlich unter Sidecar-Summe.

Push-Storm-Vermeidung

Istio: `Sidecar.egress.hosts` einschraenken, optional `ISTIO_DELTA_XDS=true` (Delta-xDS).
Linkerd: Endpoints-Slicing, kein vergleichbares Push-Problem.
Cilium: keine xDS-Pushes (Kernel/Maps), aber `cilium-agent`-CPU bei vielen Identities pruefen.

Migration & Co-Existence

Sidecar → Ambient (Istio)

Per Namespace: Sidecar-Injection aus, Label `istio.io/dataplane-mode: ambient`. Waypoint-Proxy nur deployen, wenn der Namespace L7-Policies braucht (`istioctl waypoint apply`). Migration ist inkrementell, kein Big-Bang noetig.

mTLS-STRICT-Migration

Pattern fuer alle drei: (1) mTLS `PERMISSIVE`/optional aktivieren, beide Seiten injizieren. (2) Telemetry: alle Verbindungen mTLS? (Istio: `security_policy` aus Telemetry). (3) Erst dann `STRICT`. Ohne Schritt 2 brechen Jobs, externe Health-Checks, Legacy-Clients.

Tool-Wechsel

Co-Existence im selben Cluster ist moeglich, schmerzhaft: unterschiedliche Pod-Labels, getrennte Namespaces, kein Mesh-zu-Mesh-mTLS. In der Praxis: neuen Mesh in neuem Namespace einfuehren, Workloads namespace-weise migrieren, alten Mesh am Ende entfernen.

Anti-Patterns

Was du nicht tun solltest

Mesh fuer kleine Cluster (<20 Services): Komplexitaet > Nutzen, NetworkPolicy + cert-manager reichen.
mTLS STRICT vor Telemetry-Verify: bricht Jobs, externe Health-Checks, alte Clients.
EnvoyFilter / Wasm als Default-Tool (Istio): erst Telemetry-API + AuthorizationPolicy.
EnvoyFilter bricht zwischen Istio-Minors.
Mesh-Routing auf Ingress-Layer: VirtualService an Mesh-Gateway ist ok; HTTPRoute am Ingress-Gateway + VirtualService Ost-West nicht mischen.
Tracing ohne App-Header-Propagation: Spans zerreissen.
Linkerd ohne ServerAuthorization: per-Server-Default ist allow-all – ein einfaches Pattern wie Istio's deny-all fehlt.
Cilium L7-Policy auf Multi-Tenant-Cluster ohne eBPF-Auditing: debugging schwer, Hubble-Logs aktivieren.

service-mesh-cheatsheet.de

Service Mesh Vergleich von OMNI52

Den passenden Mesh waehlen — und sauber betreiben

Architektur-Reviews & Mesh-Auswahl

OMNI52™ hilft Plattform-Teams, die richtige Service-Mesh-Architektur fuer ihren Kontext zu waehlen — und sie ohne Produktions-Riss einzufuehren.

Tool-Vergleich auf eurer Workload-Topologie, mTLS-STRICT-Migration, Multi-Cluster-Setup, Telemetry-Pipeline. Output landet als GitOps-Repo bei euch im Cluster: Helm-Charts, Runbooks, Diagnose-Skripte. Euer Team operiert weiter, nicht wir.

OMNI52 GmbH, Ebern · istio-consulting.de · kostenloses 30-min Initial-Assessment

Aus der OMNI52 Cheatsheet-Fabrik

Verwandte Sheets: istio-cheatsheet.de (Istio in der Tiefe) · kubernetes-cheatsheet.de (K8s-Plattform-Layer) · k8s-cheatsheet.de (identischer Inhalt, Kurz-Domain).

Lizenz & Weiterverteilung

Trademarks