Linkerd 2.9
mTLS for TCP, ARM support, and more!

Oliver Gould  @olix0r
Ultralight, ultrafast, security-first **service mesh** for Kubernetes.

- **4+ years in production**
- **5,000+** Slack channel members
- **10,000+** GitHub stars
- **100+** contributors
- **Weekly** edge releases
- **Open governance**, neutral home

And many more...
Two parallel branches of development:

🚀 **Linkerd 2.x**: ultralight, zero-config, Kubernetes-first (active)

🚀 **Linkerd 1.x**: JVM-based and multi-platform (maintenance)
What does Linkerd do?

⚡ **Observability:** *Golden metrics:* success rates, latencies, throughput; Service topologies; Distributed and ad-hoc tracing.

⚡ **Connectivity:** Load balancing, retries, timeouts, **multi-cluster**

⚡ **Security:** Transparent mTLS, cert management and rotation, policy*

Focused on **operational simplicity**
Microservices
Service Mesh: Data Plane
Service Mesh: Control Plane
Linkerd 2.x architecture
How is Linkerd designed?

In short, "do less, not more":

💪 **Just works:** Zero config, out of the box, for any Kubernetes app

💪 **Ultralight:** Introduce the bare minimum perf and resource cost

💪 **Simple:** Kubernetes-first; Minimal operational complexity

💪 **Security first:** Secure communication by default

**Control plane:** Go. ~200mb RSS (excluding metrics data). (Repo: [linkerd/linkerd2](https://github.com/linkerd/linkerd2)).

**Data plane:** Rust. ~20mb RSS, <1ms p99 (!!!). (Repo: [linkerd/linkerd2-proxy](https://github.com/linkerd/linkerd2-proxy))

**Background reading:** [Linkerd v2: How Lessons from Production Adoption Resulted in a Rewrite of the Service Mesh](https://www.infoq.com/articles/linkerd-lessons/) (InfoQ)
What is Linkerd's approach to security?

Linkerd is designed to enable a zero-trust approach to security. But it's easy to claim you are secure. How do you accomplish it?

- **First, do no harm.** Don't make things worse.
- **Secure the foundations.** E.g. choice of Rust for Linkerd2-proxy
- **Build on top of Kubernetes.** Don't reinvent the security wheel. (E.g.: use of ServiceAccounts for pod identity.)
- **No barrier to entry.** E.g. mTLS is on by default!
- **Keep it simple.** Complexity is the enemy of security.
What does Linkerd use for its data plane?

A purpose-built service mesh proxy, linkerd2-proxy. *Not Envoy!*

- **Security first:** Memory safety & minimal configuration surface
- **Ultralight, ultrafast:** Rust compiles to native code. No GC!
- **Audited:** Regular third-party security audits.
- **Modern async network stack:** Built on [Tokio](https://tokio.rs), [Hyper](https://hyper.rs), [H2](https://h2o.io), [Tower](https://tower.dev), and the rest of the modern Rust async networking stack for safety and performance

100% open source. 100% audited. 100% awesome! [github.com/linkerd/linkerd2-proxy](https://github.com/linkerd/linkerd2-proxy)
What Does it Do?
Peak-EWMA Load Balancing

- HTTP/1.x, HTTP/2 (gRPC), & TCP
- Efficiently distributes requests across k8s Deployments, etc
- Client-side: No centralized balancer state
- Latency-aware: Automatically optimizes for locality
- Backed by k8s Services
- ServiceTopology-aware
- Bypasses kube-proxy
- No application changes
Automatic, transparent mutual TLS

- Meshed traffic automatically secured
- Extends *workload identity* for zero-trust communication
  - Bootstrapped from k8s ServiceAccounts
- Automatic pod certificate rotation
  - Private keys never leave the pod’s memory
- Can bootstrap from [cert-manager](https://cert-manager.io)
- Does not conflict with Ingress/Application TLS
- No application changes
Transparent HTTP/2 Multiplexing

- All meshed HTTP/1.1 traffic over HTTP/2 (pod-to-pod, multi-cluster)
- Amortizes connection overhead (TCP, mTLS)
- Substantially reduces memory requirements for high-traffic sidecars
- Unique to Linkerd
- No application changes
Traffic Splitting

- For canary and blue/green
- Splits requests between k8s Services
- Uses the Service Mesh Interface’s TrafficSplit API
- Can be driven by Flagger

Now for TCP!
The Service Mesh Interface

What SMI covers

Service Mesh Interface is a specification that covers the most common service mesh capabilities:

- Traffic policy – apply policies like identity and transport encryption across services
- Traffic telemetry – capture key metrics like error rate and latency between services
- Traffic management – shift traffic between different services
Seamless, secure multi-cluster

Connects Kubernetes services across cluster boundaries in a way that’s secure, fully transparent to the application, and independent of network topology.

- **Unified trust domain** across all clusters
- **Separate failure domains** so there's no SPOF
- **Works over the open Internet** so no difficult L3/L4 requirements
- **A unified communication model** with in-cluster communication
High-fidelity Prometheus Visibility

- **Uniform**: Every pod gets the same, app-independent traffic metrics
- HTTP- and gRPC-aware
- Rich k8s workload metadata
- Raw latency histograms: no avg on latencies
- Can be enhanced with OpenAPI (Swagger) & gRPC (Protobuf) specs
- Works out-of-the-box; or *bring your own!*
- No application changes
Distributed Tracing with OpenCensus

- Linkerd participate in your application’s OpenCensus tracing
- Application changes required
Ad-hoc tracing with Linkerd Tap

- Tap into the request stream at runtime
- Authorized via k8s RBAC
- No application changes
New in 2.9.0

- Multi-arch builds for x86_64, Arm32 & Arm64
- Support for Kubernetes Service Topologies
  - Discovery now supports Kubernetes Endpoint Slices
- Bring your own Prometheus & Grafana
- Big changes to linkerd2-proxy
  - New service discovery scheme -- no more DNS dependency
  - mTLS, Load Balancing & TrafficSplit for arbitrary TCP protocols
  - More resilient HA control plane communication -- no more kube-proxy
  - Multi-threaded runtime supports scaling beyond a single CPU
  - Reduced Latency, CPU, and Memory usage
Demo Time
A brief tour of the Linkerd Lab

- **k3d** 3.2.0 (k8s 1.18)
- **linkerd** stable-2.9.0
- **ort** (oliver’s runtime tester ;)

![K3D](image1.png)
![Linkerd](image2.png)
![ORT](image3.png)
What’s the community working on?

- Minimized, modular control plane
- Multicluster routing for all TCP traffic
- Improved TCP visibility
- Bounded ServiceAccount tokens
- Traffic policy
- FIPS 140-2
- Off-cluster mesh
- Experimenting with proxy_wasm
Linkerd Community Anchor

★ Become a recognized expert
★ Tell your story in any medium
★ Submit your talk proposal with confidence
★ Get editing or writing support

Learn more on linkerd.io/community/anchor 😎
Get involved!

💚 Development is all on GitHub
💚 Thriving community in the Slack
💚 Formal announcements on the CNCF mailing lists
💚 Monthly community calls
💚 Formal 3rd-party security audits

Linkerd has a friendly, welcoming community! Join us!

Linkerd is 100% Apache v2 licensed, owned by a neutral foundation (CNCF), and is committed to open governance.