Linkerd 2.9

mTLS for TCP, ARM support, and more!

Oliver Gould @olix0r

LINKERD

Ultralight, ultrafast, security-first
service mesh for Kubernetes.

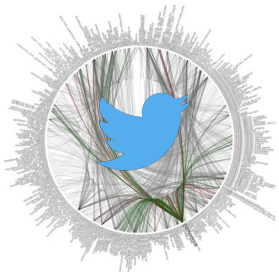
- 🔥 **4+** years in production
- 🔥 **5,000+** Slack channel members
- 🔥 **10,000+** GitHub stars
- 🔥 **100+** contributors
- 🔥 **Weekly** edge releases
- 🔥 **Open governance**, neutral home

NORDSTROM
BIGCOMMERCE
CHASE
ebay
EVERQUOTE
Expedia
OfferUp
PAYBASE
Walmart
ATTEST
Clover
cisco Webex
KAIROS
X CLOUD
finleap connect
STRAVA
Ottawa
SUBSPACE

And many more...

CLOUD NATIVE
COMPUTING
FOUNDATION

History of Linkerd



LINKERD 0.1

CLOUD NATIVE COMPUTING FOUNDATION

LINKERD 2.0

2010-2015 | 2/2016 | 1/2017 | 9/2018

Two parallel branches of development:

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

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

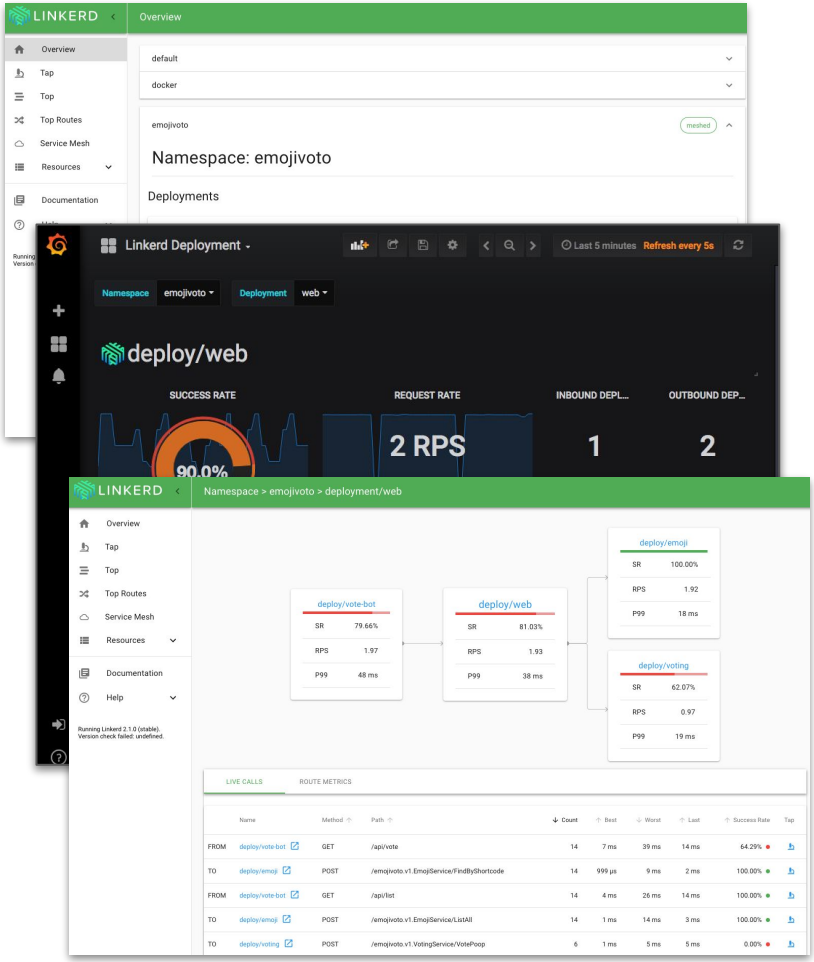
What does Linkerd do?

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

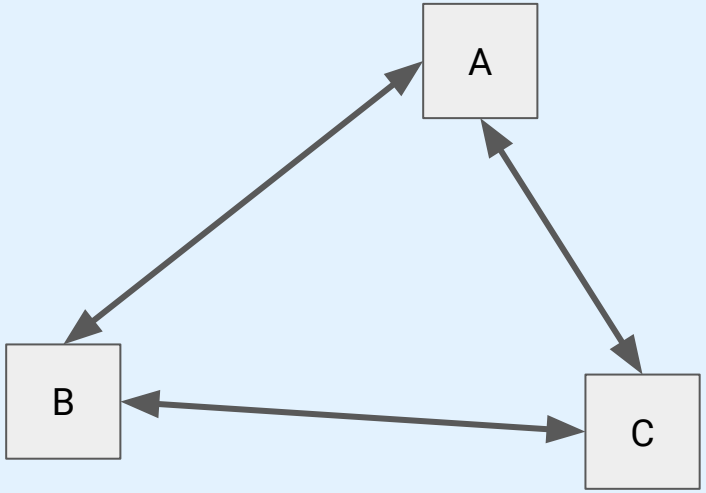
⚡ **Connectivity:** Load balancing, retries, timeouts, **multi-cluster**

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

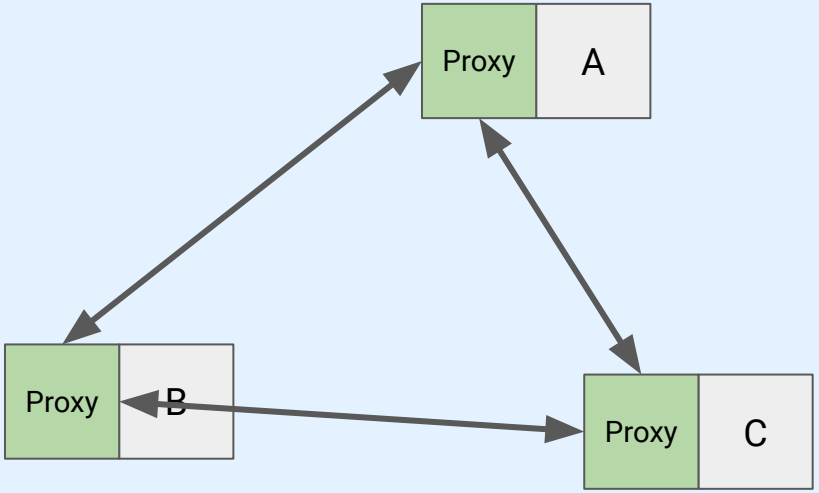
Focused on **operational simplicity**



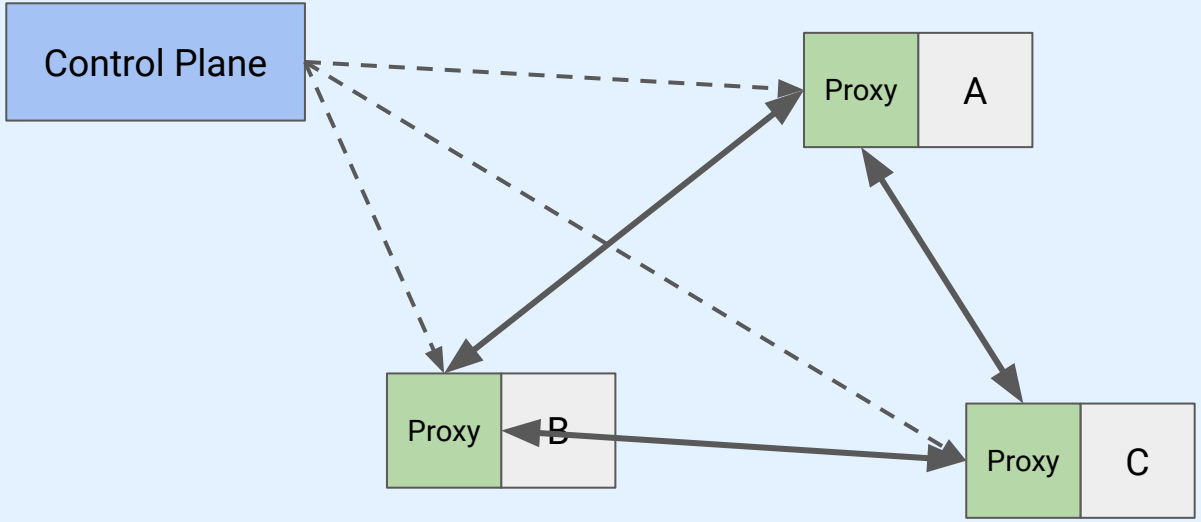
Microservices



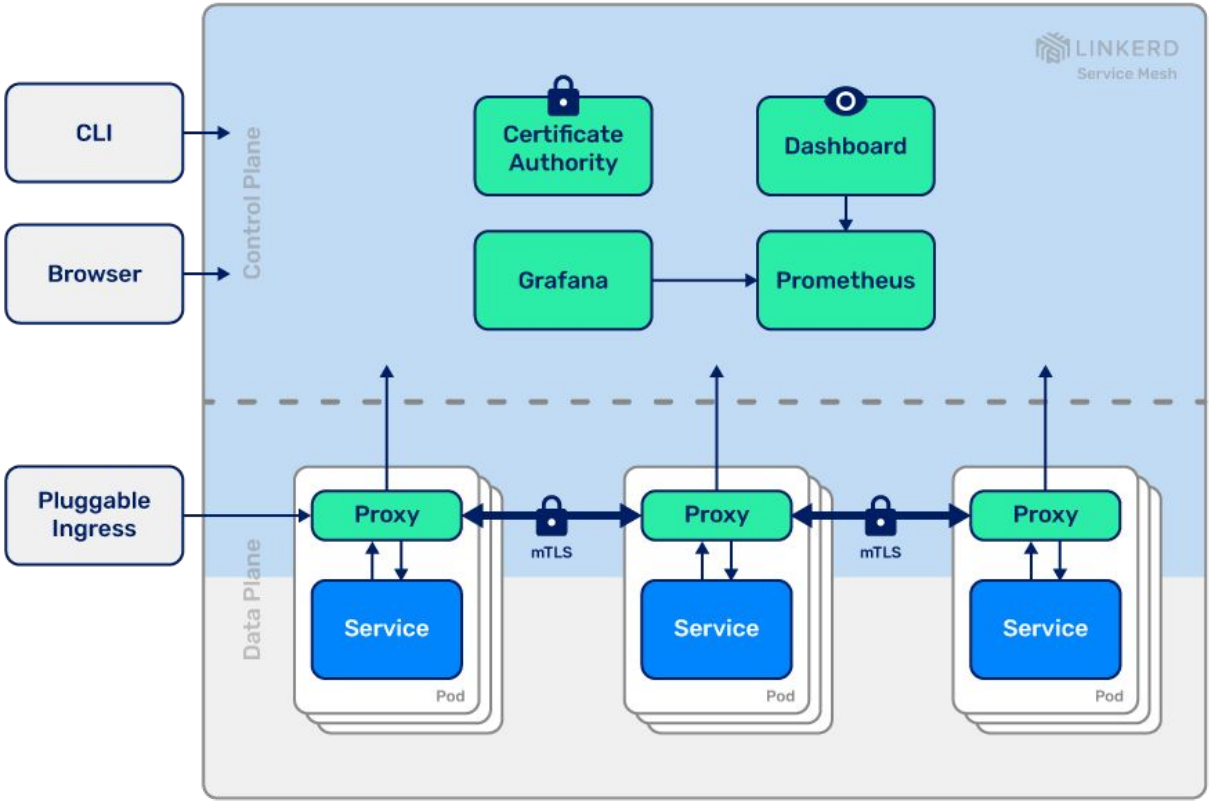
Service Mesh: Data Plane



Service Mesh: Control Plane



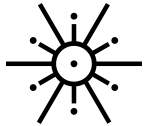
Linkerd 2.x architecture



kubernetes

GOLANG

Service Mesh Interface



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).

Data plane: Rust. ~20mb RSS, <1ms p99 (!!!). (Repo: linkerd/linkerd2-proxy)

Background reading: Linkerd v2: How Lessons from Production Adoption Resulted in a Rewrite of the Service Mesh (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.

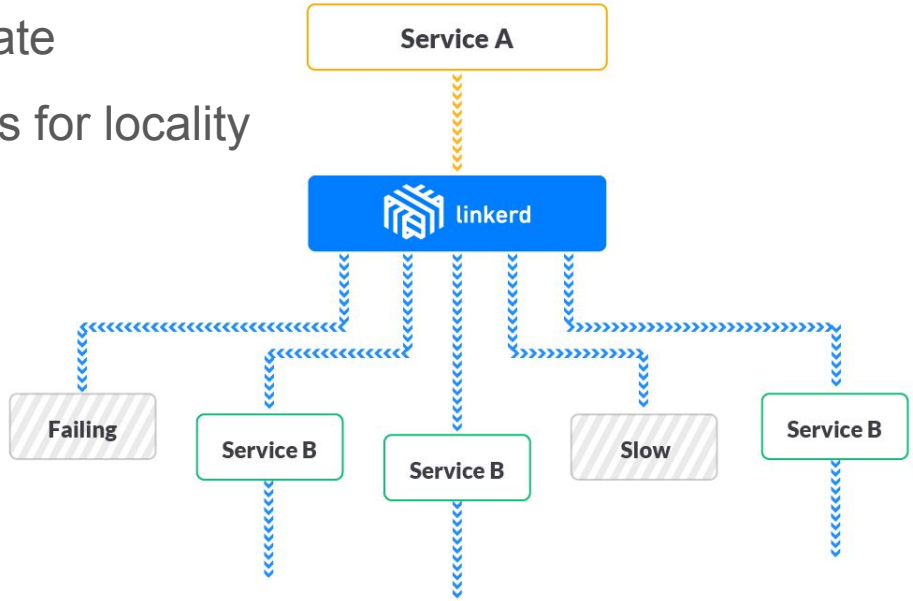
🚀 **Modern async network stack:** Built on Tokio, Hyper, H2, Tower, and the rest of the modern Rust async networking stack for safety and performance

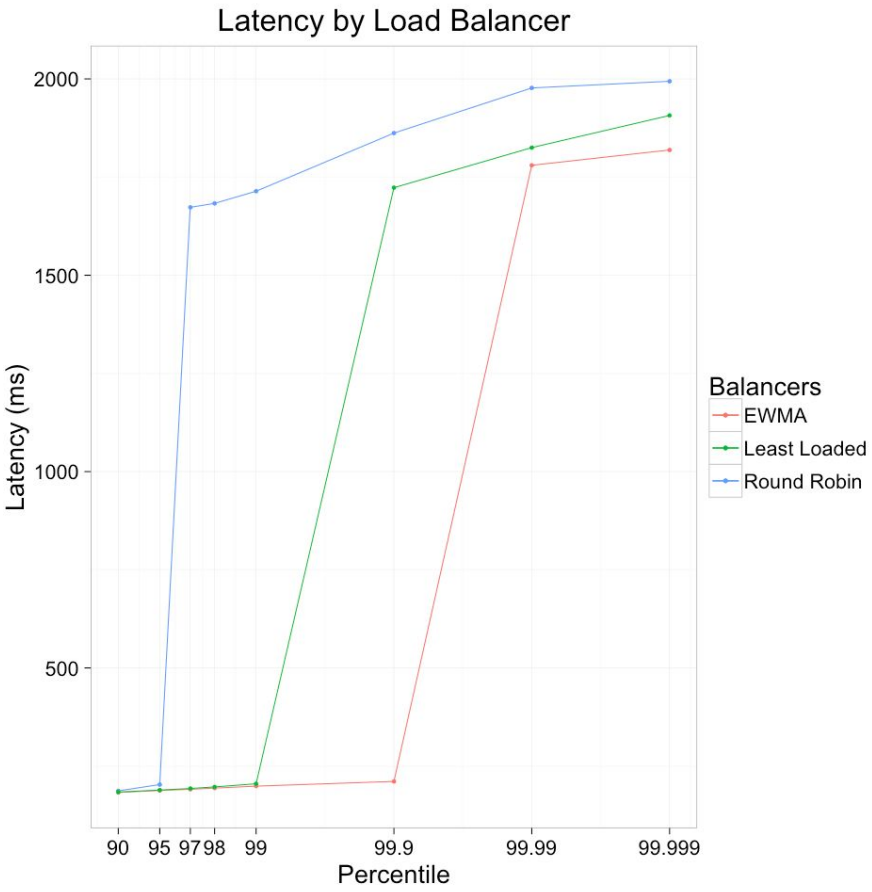
100% open source. 100% audited. 100% awesome! github.com/linkerd/linkerd2-proxy

What Does it Do?

Peak-EWMA Load Balancing

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



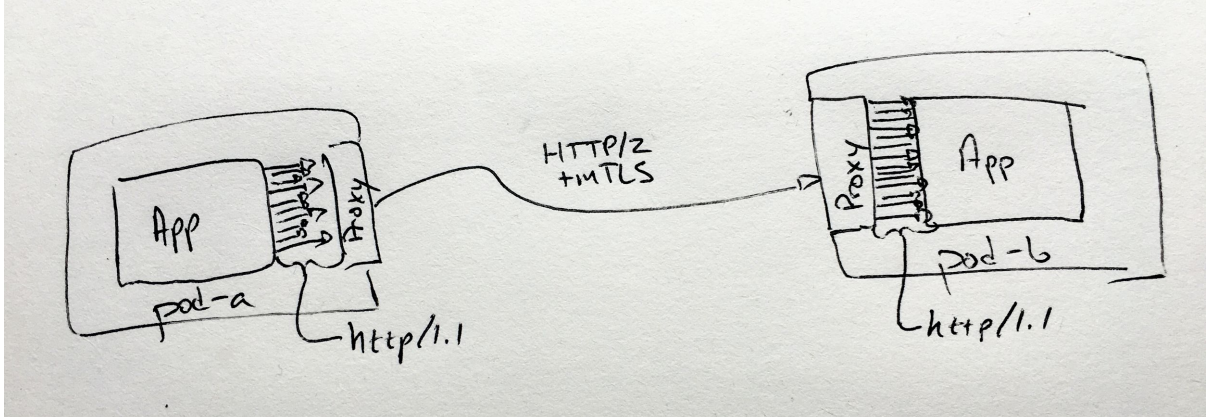

Latency by Load Balancer
Latency (ms)
2000
1500
1000
500
90
95
97
98
99
99.9
99.99
99.999
Percentile
Balancers
EWMA
Least Loaded
Round Robin

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`
- Does not conflict with Ingress/Application TLS
- No application changes

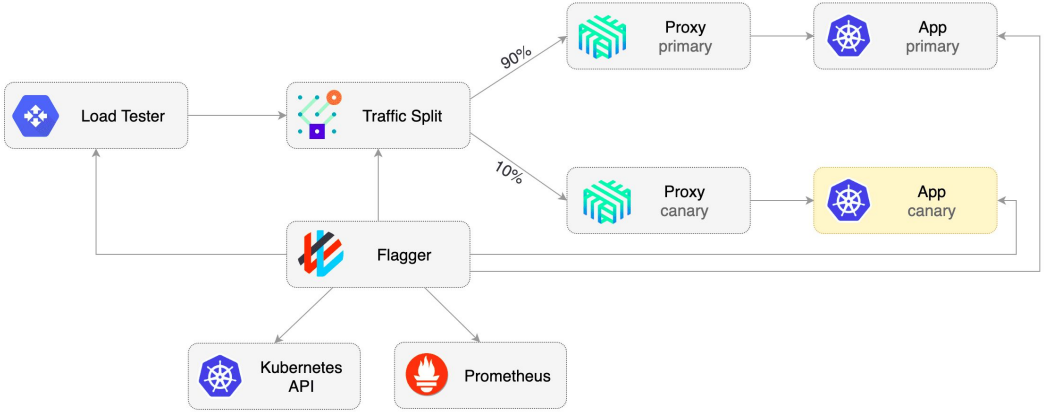
Transparent HTTP/2 Multiplexing

- 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

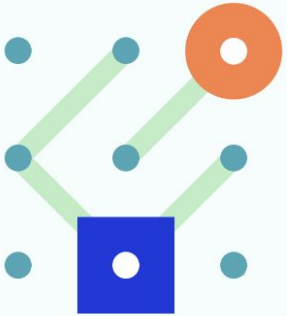


Traffic Splitting

- For canary and blue/green
- Splits requests between k8s `Services`
- Uses the [Service Mesh Interface](#)'s `TrafficSplit` API
- Can be driven by [Flagger](#)



The Service Mesh Interface



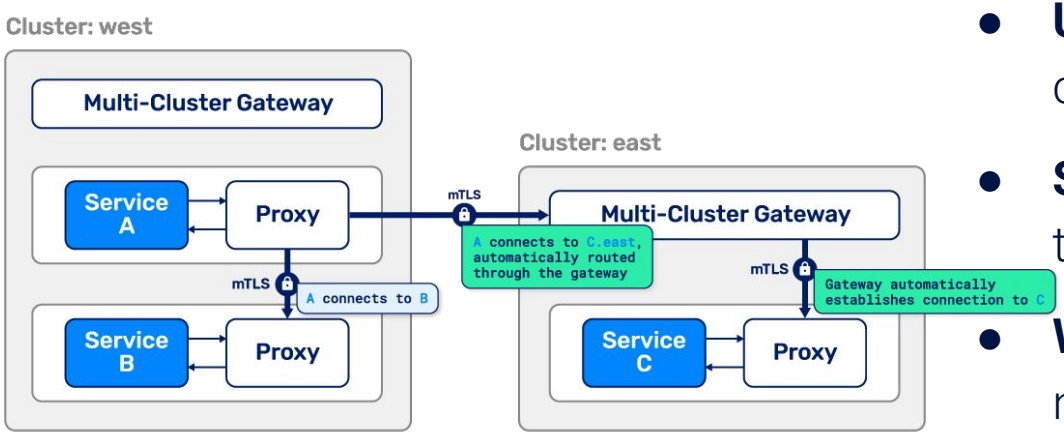
What SMI covers

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.



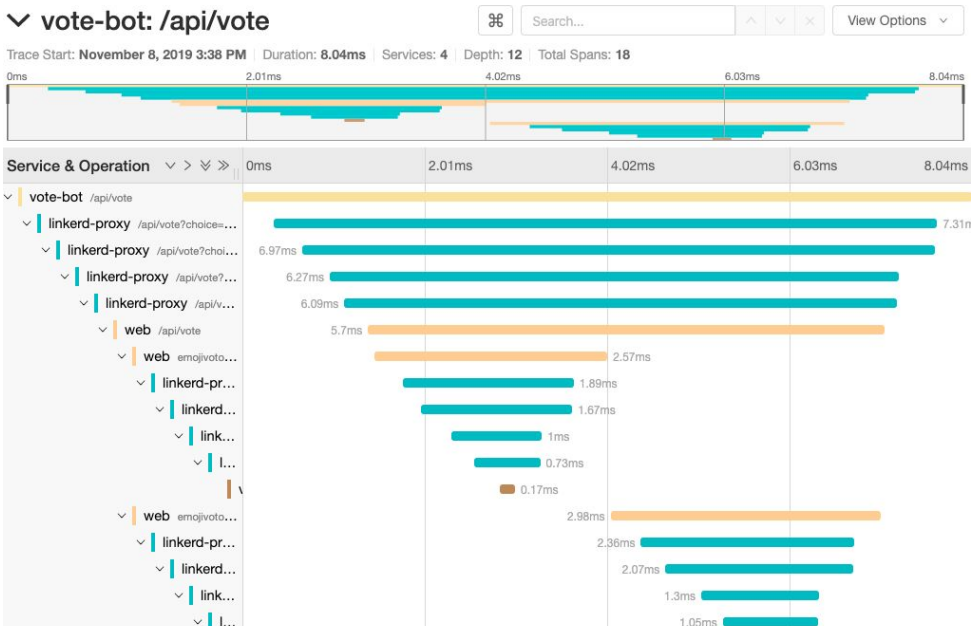
- **Unified trust domain** across all clusters
- **Separate failure domains** so there's no SPOF
- **Works over the open Internet** so no difficult L3/L4 requirements
- **A unified communication model** with in-cluster communication

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

```
(press q to quit)
(press a/LeftArrowKey to scroll left, d/RightArrowKey to scroll right)
```

Source	Destination	Method	Path	Count	Best	Worst	Last	Success Rate
linkerd-prometheus-5dd896954c-g7snn	10.244.0.219	GET	/metrics	6	1ms	3ms	2ms	100.00%
linkerd-prometheus-5dd896954c-g7snn	10.244.4.222	GET	/metrics	5	2ms	3ms	2ms	100.00%
linkerd-prometheus-5dd896954c-g7snn	10.244.1.16	GET	/metrics	5	2ms	3ms	2ms	100.00%
linkerd-prometheus-5dd896954c-g7snn	10.244.4.221	GET	/metrics	5	1ms	4ms	3ms	100.00%
linkerd-prometheus-5dd896954c-g7snn	10.244.2.82	GET	/metrics	5	2ms	4ms	4ms	100.00%
linkerd-prometheus-5dd896954c-g7snn	10.244.3.116	GET	/metrics	4	1ms	3ms	1ms	100.00%
linkerd-prometheus-5dd896954c-g7snn	10.244.3.115	GET	/metrics	4	1ms	3ms	3ms	100.00%
10.244.4.1	linkerd-grafana-548d67bdd-ftv62	GET	/api/health	4	448µs	547µs	530µs	100.00%
linkerd-prometheus-5dd896954c-g7snn	10.244.4.220	GET	/metrics	4	2ms	4ms	4ms	100.00%
linkerd-prometheus-5dd896954c-g7snn	10.244.0.220	GET	/metrics	3	1ms	1ms	1ms	100.00%
10.244.2.1	linkerd-destination-6d9d9dfbf6-fq6hd	GET	/ready	3	395µs	629µs	395µs	100.00%
linkerd-prometheus-5dd896954c-g7snn	linkerd-sp-validator-77f8b989-g6bjq	GET	/metrics	3	2ms	2ms	2ms	100.00%
10.244.3.1	linkerd-web-55bfcf9698-5wxwf	GET	/ping	3	449µs	723µs	472µs	100.00%
linkerd-prometheus-5dd896954c-g7snn	linkerd-controller-78844b9b87-z8sgl	GET	/metrics	3	2ms	2ms	2ms	100.00%
linkerd-prometheus-5dd896954c-g7snn	linkerd-grafana-548d67bdd-ftv62	GET	/metrics	3	2ms	3ms	2ms	100.00%
linkerd-prometheus-5dd896954c-g7snn	linkerd-destination-6d9d9dfbf6-fq6hd	GET	/metrics	3	2ms	3ms	3ms	100.00%
10.244.0.1	linkerd-sp-validator-77f8b989-g6bjq	GET	/ready	3	466µs	885µs	573µs	100.00%
linkerd-prometheus-5dd896954c-g7snn	linkerd-proxy-injector-648d6864b6-f8fqt	GET	/metrics	2	2ms	2ms	2ms	100.00%
10.244.1.1	linkerd-controller-78844b9b87-z8sgl	GET	/ping	2	346µs	578µs	578µs	100.00%
10.244.3.1	linkerd-web-55bfcf9698-5wxwf	GET	/ready	2	453µs	614µs	453µs	100.00%
10.244.1.1	linkerd-prometheus-5dd896954c-g7snn	GET	/-/healthy	2	459µs	468µs	468µs	100.00%
linkerd-prometheus-5dd896954c-g7snn	linkerd-web-55bfcf9698-5wxwf	GET	/metrics	2	2ms	2ms	2ms	100.00%
10.244.3.1	linkerd-proxy-injector-648d6864b6-f8fqt	GET	/ping	2	461µs	490µs	490µs	100.00%
10.244.0.1	linkerd-sp-validator-77f8b989-g6bjq	GET	/ping	2	375µs	532µs	375µs	100.00%
10.244.1.1	linkerd-controller-78844b9b87-z8sgl	GET	/ready	2	432µs	446µs	432µs	100.00%
10.244.1.1	linkerd-prometheus-5dd896954c-g7snn	GET	/-/ready	2	646µs	668µs	668µs	100.00%
10.244.2.1	linkerd-destination-6d9d9dfbf6-fq6hd	GET	/ping	2	537µs	614µs	614µs	100.00%
10.244.3.1	linkerd-proxy-injector-648d6864b6-f8fqt	GET	/ready	2	602µs	969µs	602µs	100.00%

New in 2.9.0

- Multi-arch builds for x86_64, Arm32 & Arm64
- Support for Kubernetes ServiceTopologies
  - Discovery now supports Kubernetes EndpointSlices
- 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 ;)

Looking Forward

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.

Cole Calistra @coleca · Feb 2
FACT: If you are considering service mesh and @linkerd isn't first on your list you're making a HUGE mistake. It just WORKS. Plain and simple. No hours of YAML configuration files to write. It just WORKS. Thank you @wm and @BuoyantIO team! @CloudNativeFdn

Site Reliability Balladeer @SethMcCombs · 8 Dec 2018
Replying to @michellenoorali
It took me a total of 5 minutes to set up @linkerd in my QA environment and BOOM metrics for days. I can't remember the last time I set up something so easy, it was almost...fun?

ZΔK @zakknill · Feb 14
Just used #linkerd2 for the first time to solve a real production issue. The observability tooling is life changingly good! Thanks @linkerd

Abhinav Khanna @Abhinav14435957 · 12 Dec 2018
Having used Linkerd, I think the team has done a fantastic job of making it feel magical. #linkerd

Michelle Noorali @michellenoorali · 8 Dec 2018
seriously the linkerd2 getting started guide is so good and the check command is just beautiful 😘 linkerd.io/2/getting-star... @linkerd

Nigel Wright @nigelwright_nz · 18 Nov 2018
Whoa @linkerd just blew my mind a little. That was crazy easy to setup and start getting real info about my #k8s deployments.

Stephen Pope @stephenpope · 26 Oct 2018
@linkerd Very pleased with #Linkerd2 - deployed my app (with auto-proxy-injection) and #itjustworked - Had all the info I needed on the dashboard - Thanks very much (great docs too)

Darren Shepherd @ibuildthecloud · Feb 14
I'm consistently impressed with @linkerd 2.0. If you are looking at istio, try linkerd first. I takes about 5 minutes. Then you'll have something working and in place while you try to understand and deploy istio for the next 9 months.

9 63 270