BUOYANT

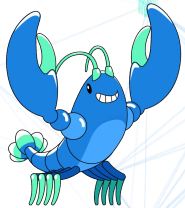
Creators of LINKERD



Metrics and Dashboards and Charts, oh my!

Flynn

Technical Evangelist for Linkerd



O'REILLY®

BUOYANT
Creators of LINKERD

Linkerd: Up & Running

A Guide to Operationalizing a Kubernetes-Native Service Mesh

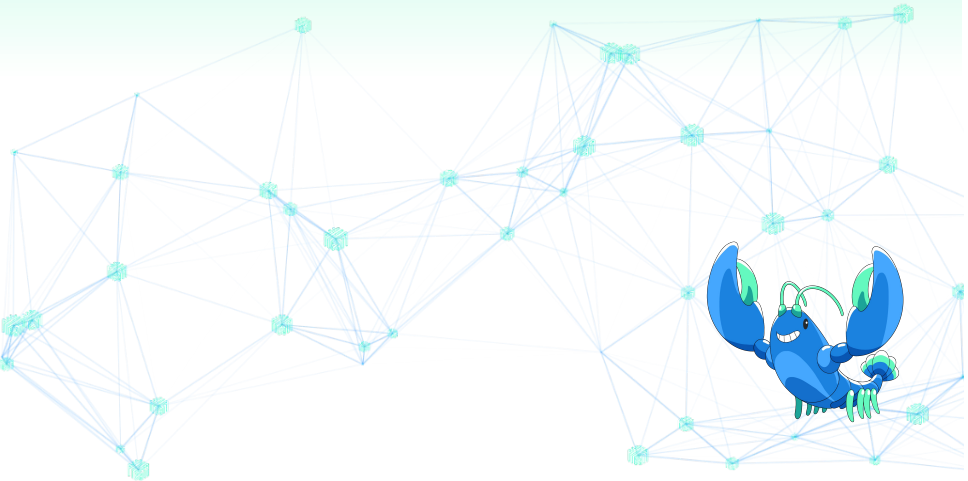


Download for free!

buoyant.io/**download/linkerd-up-and-running**

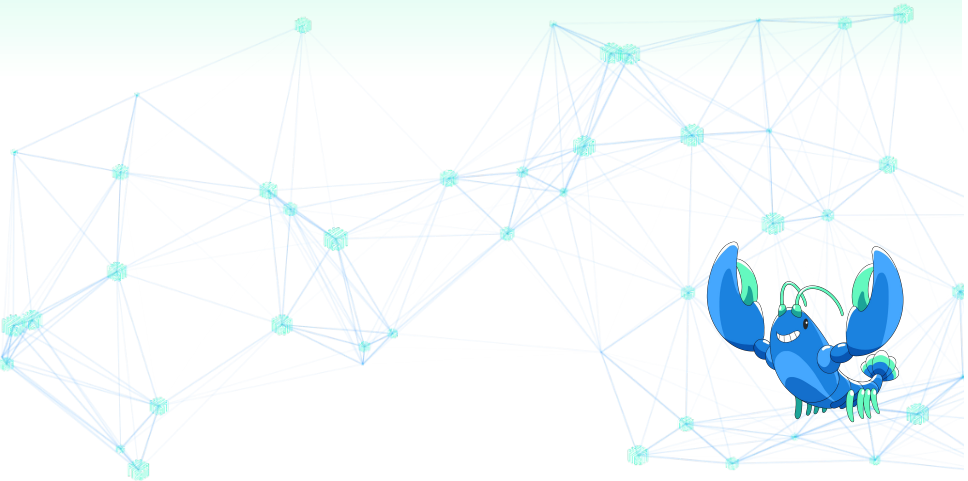
What's on the agenda?

- **Discussion** about metrics,
- **Demo** but we'll interleave that with more about the metrics as we go!



BUOYANT

How do you follow along?



- https://github.com/BuoyantIO/service-mesh-academy/tree/main/metrics-dashboards-charts
- I'll be using **Buoyant Enterprise for Linkerd `2.16.0`**, but Linkerd **`edge-24.8.2`** or later should be fine.
- I'll be using k3d but you do you. https://k3d.io/
- You'll want `promtool` from the Prometheus installation too. https://prometheus.io/download/

BUOYANT

BUOYANT
SERVICE MESH
ACADEMY

Metrics



Definitions

Metrics, *n*:

- Information that's helpful **for you** to better understand what's going on with your stuff
- The Golden Metrics (request rate, success rate, and latency) are a well-known example
- There are **many** others

I'm making a distinction here between *metrics* and *telemetry*:

- Telemetry is about information flowing to **someone else**; metrics is about information **for you**.

BUOYANT
Creators of LINKERD

The First Law of Metrics

- **You have to know what you need.**

There's no one-size-fits all metrics dashboard or monitoring system out there.

Understand your application, use that to understand what things you need to be able to see, then put the pieces together to see those things.

A Whirlwind Overview of Metrics

Metrics are often stored in a **time series database**.

- Nonrelational database: “here’s a key, give me the data you have for it.”
- Relational database: “here’s a relationship, what data satisfies it?”
- Time-series database: “here’s a metric, give me a bunch of data points showing me how that metric is changing over time.”

This implies that time series databases don’t overwrite samples, they just append new ones.

BUOYANT
Creators of LINKERD

A Whirlwind Overview of Metrics

There are several time-series databases out there.

- RRDTool was the original (1999!).
- Many relational databases can (at least sort of) do time series.
- Prometheus is the major CNCF player.

Linkerd uses Prometheus.

Of course, every TSDB has its complicating factors, and Prometheus is no different. 😀

Complicating Factor 1: Dimensions

Each metric is stored across time **and** across multiple *dimensions*

- Really a dimension is just another identifying label
  - not just “success rate”, but “success rate for deployment X talking to Service Y”
  - `deployment`=“X” and `Service`=“Y” are *labels*
  - `deployment` and `Service` are *dimensions*
- You write `metric_name{label1=“value1”, label2=“value2”}` if you want to restrict which dimensions you care about.
- Writing just `metric_name` means that you want **all** the dimensions for that metric.

BUOYANT
Creators of LINKERD

Complicating Factor 1: Dimensions

In general, you want to be careful about how many different dimensions you're working with.

- The TSDB folks talk about *cardinality* as a measure of how much data you have across how many dimensions, so you may run across that term being kicked around.
- For our purposes as **users** of metrics, though, it has more to do with not being overwhelmed by labels for things you don't care about.
- More on this in the demo!

Complicating Factor 2: Data Types

There are different types of metrics, and the type constrains what you can do with the data. Three common types:

- **gauge**: a number that can go up and down
  - example: the number of endpoints blocked by circuit breaking
- **counter**: a number that can only go up
  - example: the number of HTTP requests that have succeeded
- **histogram**: a number that will be grouped into a bunch of buckets for later analysis
  - example: the latency of a request

BUOYANT
Creators of LINKERD

Complicating Factor 3: Queries

Generally speaking, when you write code to generate metrics, you pick the data type that's the fastest and simplest to generate, because you can't slow down your code too much.

Sadly, this is rarely the most useful thing when it comes to *using* the metrics later. 😂

For example, in general no one stores a "success rate" metric.

- Instead, you store a counter for total requests and a counter for successful requests…
- …or a counter for "requests" that includes a label for the request status.

Complicating Factor 3: Queries

Linkerd uses the second option:

- `response_total` is a counter of HTTP responses
- It has a `classification` label (e.g. `classification="success"`)
- (it has a lot of other labels too, more on this in a bit)

Of course this means you can't just query for success rate.

Complicating Factor 3: Queries

Instead:

- Figure out how fast `response_total{classification="success"}` is rising
- Figure out how fast `response_total` (irrespective of `classification`) is rising
- Divide

Oh, and make sure to get all the other labels correct while you do this. 😀

BUOYANT
Creators of LINKERD

Complicating Factor 3: Queries

Rather than duplicating the math every time we want to use math, it's generally built into the database, so you can make queries like

```
sum(rate(response_total{classification="success", other-labels}[1m]))
    by (label)
  /
sum(rate(response_total{other-labels}[1m]))
    by (label)
```

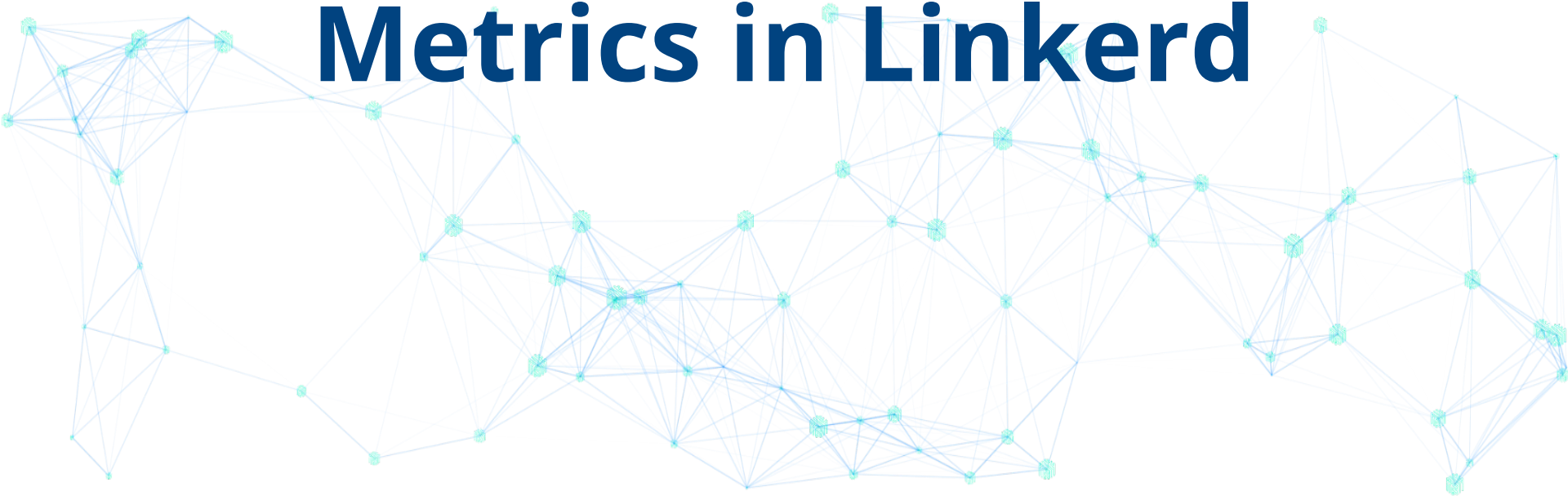
- `rate` calculates a rate from a counter (over 1 minute here)
- `sum by (label)` is really a way to say "collapse all the dimensions except for *label*"

Don't panic. It's manageable. 😀

BUOYANT
Creators of LINKERD

BUOYANT
SERVICE MESH
ACADEMY

Metrics in Linkerd



Metrics in Linkerd

- Provided by the control plane and the data plane
- Stored in the Linkerd pods in a Prometheus-compatible way
- Accessible on port 4191 of any Linkerd pod, so that Prometheus can pull metrics as needed ("scraping")
- Also accessible via `linkerd diagnostics controller-metrics` and `linkerd diagnostics proxy-metrics` in raw form…
- …or via Linkerd Viz, which is really just a metrics dashboard.

Metrics in Linkerd

Let's repeat that:

- The Linkerd Viz dashboard just displays metrics from a Prometheus instance.
- It has no special privileges or anything; it's just a Prometheus dashboard tailored to Linkerd.
- Feel free to write your own. 😀

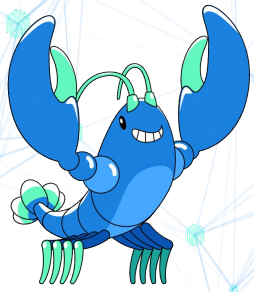
Metrics in Linkerd

Storage is a **critical concern here**

- The Prometheus that ships with Linkerd Viz is **not acceptable for real use**
- In the real world, you'll need to think carefully about how long you want to preserve metrics, whether there are dimensions you can drop when scraping, etc.
- This is **far** beyond the scope of this talk, but it is very, very important.

BUOYANT
SERVICE MESH
ACADEMY

DEMO 🤞



BUOYANT
SERVICE MESH
ACADEMY

Linkerd Metrics

(as opposed to metrics in Linkerd 🙄)

“Control” metrics

Names starting with `control_` are Linkerd control plane metrics.

- We’re not going to talk too much about these; they’re the kind of thing that’s typically most interesting if a Linkerd maintainer is trying to help you debug something weird.
- (Sort of) exceptions:
  - `control_identity_cert_*` will show you things about your workload certificates rotating
  - `control_request_*` and `control_response_*` show you things about your workloads talking to the control plane

BUOYANT
Creators of LINKERD

“Inbound” metrics

Names starting with `inbound_` are metrics about inbound requests.

- `inbound_http_authz_allow_total` and `inbound_http_authz_deny_total` track authorization success and failure
- `inbound_http_errors_total` tracks errors in inbound HTTP
  - the “`error`” label has a reason (e.g. “`i/o`”)

These are all counters, so you’ll most likely want to use `rate()` with them.

BUOYANT
Creators of LINKERD

“Outbound” metrics

Names starting with `outbound_` are metrics about outbound requests.

- `outbound_http_*` metrics track HTTP requests
- `outbound_grpc_*` metrics track gRPC requests
  - **YOU MUST DEFINE GRPCRoute RESOURCES TO GET gRPC METRICS.**
  - Linkerd will synthesize “default” routes internally for HTTP, but it **won’t** do that for gRPC.

BUOYANT
Creators of LINKERD

“Outbound” metrics

`outbound_http_route_backend_requests_total`
`outbound_grpc_route_backend_requests_total`

- These are the main metrics for tracking outbound requests by route.
- They identify the source with the `deployment` and `namespace` labels.
- The `backend_*` labels identify backendRefs (`backend_kind`, `backend_group`, `backend_name`, and `backend_namespace`)
- The `parent_*` labels identify parentRefs (`parent_kind`, `parent_group`, `parent_name`, and `parent_namespace`)

BUOYANT
Creators of LINKERD

“Outbound” metrics

`outbound_http_route_backend_requests_total`
`outbound_grpc_route_backend_requests_total`

- Synthesized routes have `route_kind=“default”` and `route_name=“http”`
  - Remember, you won’t see these for gRPC!
- Real routes have `route_kind=“HTTPRoute”` or `“GRPCRoute”`, `route_group=“gateway.networking.k8s.io”`, and `route_name`=the name of the HTTPRoute or GRPCRoute.

BUOYANT
Creators of LINKERD

“Outbound” metrics

`outbound_http_route_backend_response_statuses_total`

- Like `outbound_http_route_backend_requests_total`, but includes an `http_status` label that’ll have the HTTP status code
  - e.g. `http_status=“200”`
  - Timeouts will also have `error=“REQUEST_TIMEOUT”` or `error=”REQUEST_HEADERS_TIMEOUT”`

“Outbound” metrics

`outbound_grpc_route_backend_response_statuses_total`

- Like `outbound_grpc_route_backend_requests_total`, but includes an `grpc_status` label that’ll have the gRPC status.
  - e.g. `grpc_status=“OK”`

`outbound_http_route_request_statuses_total`
`outbound_grpc_route_request_statuses_total`

- Aggregate across all backends rather than just being for a single backend

BUOYANT
Creators of LINKERD

“Outbound” metrics

`outbound_http_route_retry_requests_total`

- Retry metrics! This is the total number of retries sent.

`outbound_http_route_retry_successes_total`

- Total number of retries that succeeded

`outbound_http_route_retry_limit_exceeded_total`

- Number of retries that we gave up on because we hit the maximum retry count

`outbound_http_route_retry_overflow_total`

- Number of retries that we didn’t send because a circuit breaker was open

BUOYANT
Creators of LINKERD

“Balancer” metrics

Names containing with `_balancer_` are metrics about the proxy’s outbound load balancers.

- The load balancer is the thing that chooses which endpoint to talk to for a given request.
- There’s a ton of stuff buried in these metrics but for our purposes here, the really critical one is `outbound_http_balancer_endpoints{endpoint_state="pending"}`
  - This reflects the number of endpoints for which a circuit breaker is open.

BUOYANT
Creators of LINKERD

Aggregated request/response metrics

Names starting with `request_` or `response_` are non-route-aware aggregators for request and response information.

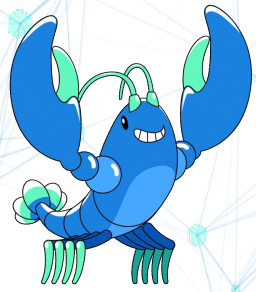
- These also lump together inbound and outbound (using the `direction` label)
- In general, you probably want to use dedicated `inbound_*` or `outbound_*` metrics if you can
  - it's likely going to be easier on Prometheus

BUOYANT
Creators of LINKERD

BUOYANT
SERVICE MESH
ACADEMY

back to the DEMO





Always Remember The First Law of Metrics

- **You have to know what you need.**

Linkerd publishes some canned dashboards and they're very helpful, but you really do need to know what you want and be ready to put together exactly the things that you need.

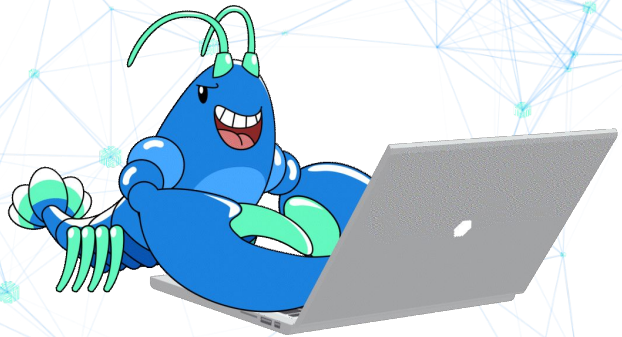
This sounds much scarier than it actually is. 🙂

SERVICE MESH ACADEMY

Monthly hands-on, engineer-focused training from the creators of the service mesh

September 12: The Top Five Things You Need to Know About Linkerd 2.16

SIGN UP TODAY!
buoyant.io/sma



Buoyant Enterprise for LINKERD

Rust-based network security and reliability for modern applications. Built on open source and designed for the enterprise.

- ➔ Zero-trust security and compliance across your entire network
- ➔ Global traffic management and control
- ➔ Full L7 application observability
- ➔ Built for the enterprise

Learn more & get a free trial at buoyant.io/**enterprise-linkerd**

BUOYANT
Creators of LINKERD

BUOYANT
SERVICEMESH
ACADEMY

Q&A



BUOYANT
SERVICEMESH
ACADEMY

Thanks much!



flynn@buoyant.io
@flynn on slack.linkerd.io