

# Debugging services in Kubernetes with Linkerd 2.0

**Risha Mars**  
**@marzipan**  
**[mars@buoyant.io](mailto:mars@buoyant.io)**



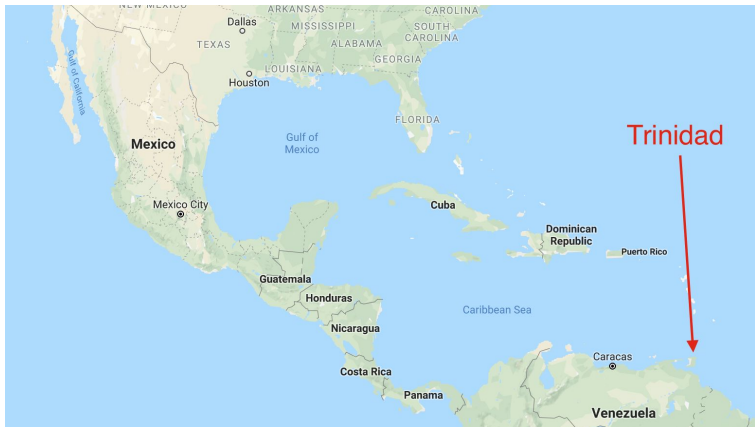
**BUOYANT**

# Risha Mars (me. hi!)

Software Engineer at Buoyant

✉️ mars@buoyant.io

🐦 @marzipan



# The Plan



Observe a demo application failing



Try to diagnose the issue using tools we know



Intro to Linkerd

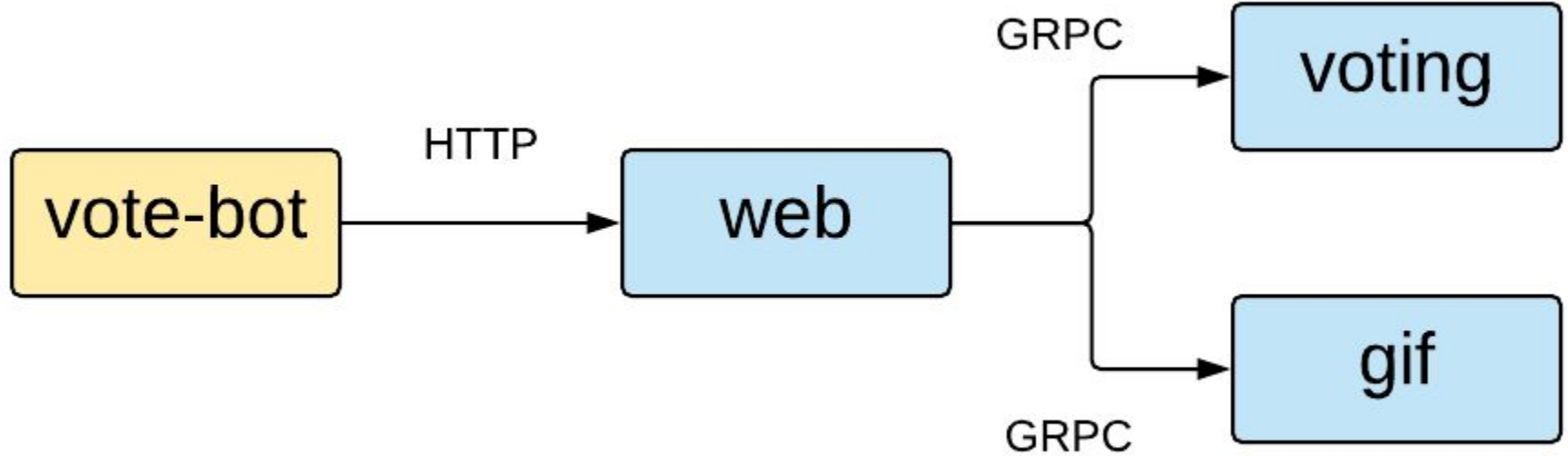


Install Linkerd



Diagnose issue!

# My application: nodevoto



## Server on fire!

Today, 2:26 PM

Assigned to: Rachael Byrne

Service: PDT-RB

Incident Details

Timeline

### SUBJECT

Server on fire!

### DETAILS

### INCIDENT SUMMARY

Server on fire!

  
ACK

  
RESOLVE

  
MORE



**Honest Status Page**

@honest\_update

Follow



We replaced our monolith with micro services so that every outage could be more like a murder mystery.

4:10 PM - 7 Oct 2015

# How do we solve this mystery?

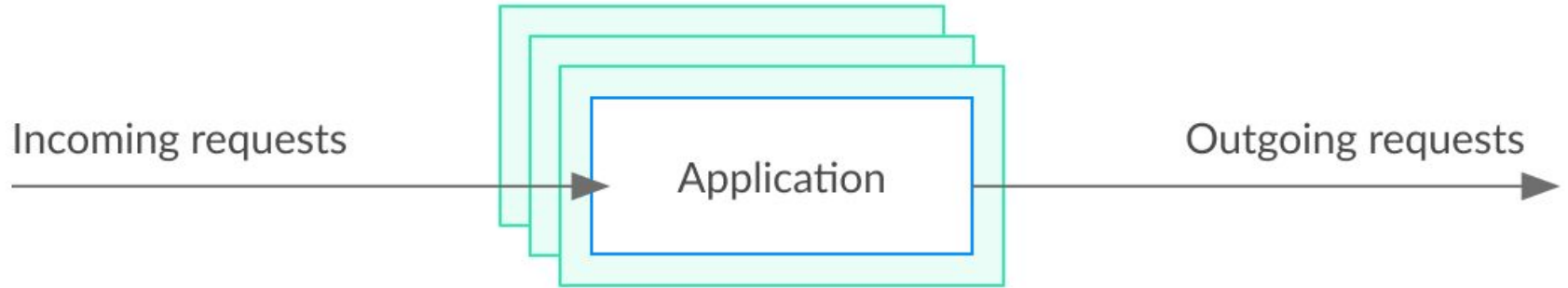
- Improve our logging
  - What if it's a service we don't control?
- We need visibility into our app!
  - Add instrumentation libraries to the code
  - Add monitoring (e.g. Prometheus)
  - Set up dashboard (e.g. Grafana)
- Is there an easier way?

The background is a solid blue color with several diagonal stripes of varying shades of blue and cyan running from the bottom-left towards the top-right. The stripes are of different widths and are semi-transparent, creating a layered effect.

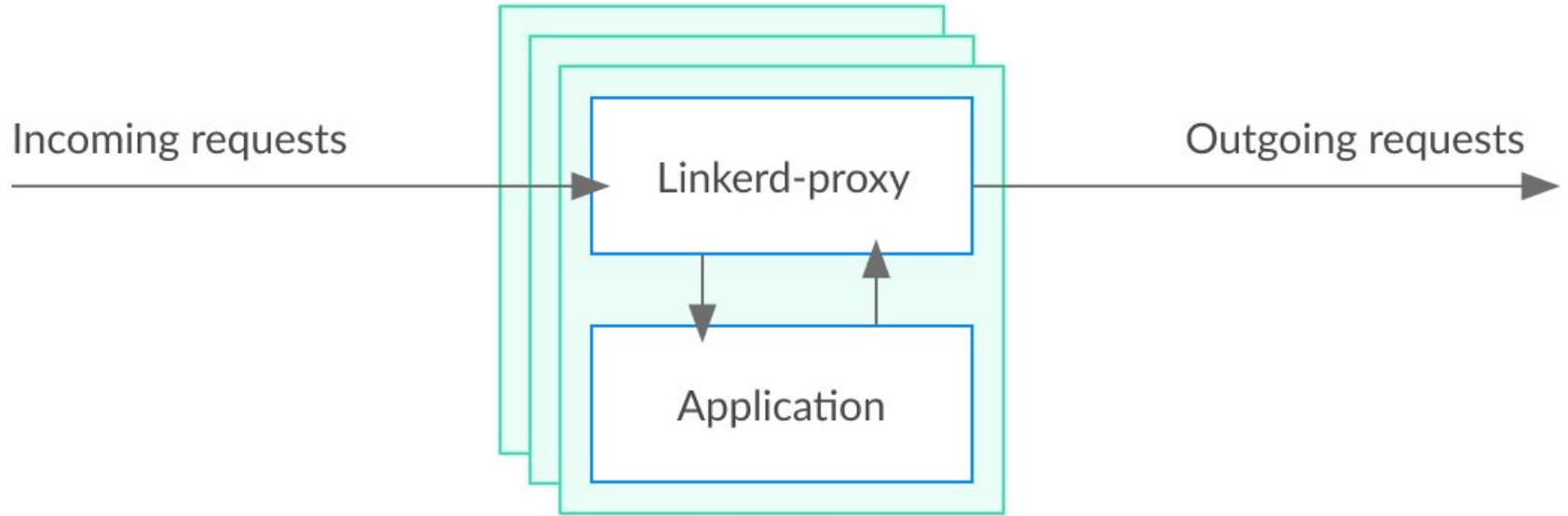
**Linkerd**



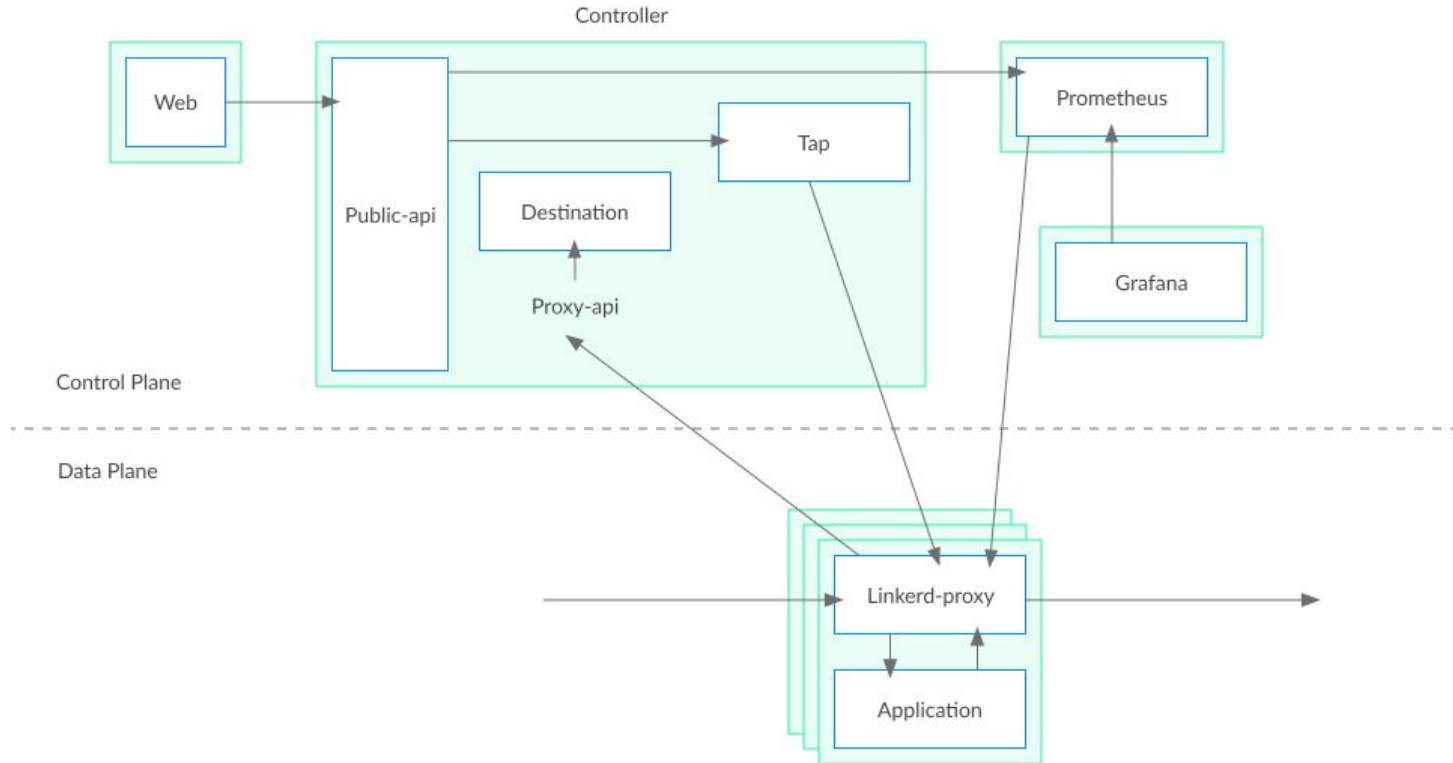
# Your application



# Your application + data plane




# Linkerd 2.0 architecture



# Mystery solved!



# What did we do?

- Installed linkerd control plane
- Added web to the linkerd data plane
- Got the observability we needed! 
  - Used dashboard to find problematic endpoint
  - Inspected Grafana dashboards
  - Viewed live requests through the system

# What did we NOT do?

- No application code changes!
- No config changes!
- No bothering of other service owners!
- No bothering of ops / platform owners!
- No complicated wiring up of components!
  - No having to wire up Grafana dashboards ourselves!
  - No having to configure our own Prometheus!

# Features

- Latency-aware load balancing
- Service Profiles (per route metrics)
- Retries and timeouts
- Ingress
- TCP proxying, protocol detection
- Telemetry, monitoring
- HTTP, HTTP/2, gRPC proxying
- Automatic TLS

# **Try it yourself!**

<https://bit.ly/linkerd-get-started>





Currently, two parallel branches of development:

- Linkerd 1.x: powerful, highly configurable, multi-platform (K8s, ECS, Mesos, Consul/Nomad)
- Linkerd 2.x: ultralight, zero-config, Kubernetes-only.

<https://github.com/linkerd>



24+ months in production

2k+ Slack channel members

7,000+ GitHub stars

20m+ DockerHub pulls

80+ contributors

400b+ production requests/mo



credit karma



<http://github.com/linkerd>



# Contribute!

<http://github.com/linkerd>

Golang, Javascript, Rust, Scala

# Recap time!

- What do we need to have insight into our service?
- What is a Service Mesh?
  - Linkerd overview: data plane, control plane
- Use Linkerd's features to diagnose issues
  - Install Linkerd
  - Add Linkerd to our service
- Other cool Linkerd features
  - per-route metrics, TLS, load balancing, retries, timeouts

# Learn more!

Slides: <https://github.com/rmars/talks>

Try it yourself!

- <https://bit.ly/linkerd-get-started>
- <https://docs.linkerd.io>

Ask questions!

- <https://slack.linkerd.io/>

Contribute!

- <https://github.com/linkerd/linkerd2>

Risha: [mars@buoyant.io](mailto:mars@buoyant.io) / @marzipan

**Questions!?**

# Linkerd 2.x design goals

- 🔧 **Zero-config “just works”**. It should work with any K8s app
- 🔧 **Really, really, really fast**: proxies should introduce the bare minimum perf (and resource!) hit
- 🔧 **Understandable**: no magic.

**Data plane:** [linkerd2-proxy](#). Written in Rust. < 10ms RSS, < 1ms p99. (!!!!)

**Control plane:** [linkerd2](#). Written in Go. Includes small Prometheus (6 hour window), Grafana, etc.