

# Building Service Ownership Using Documentation, Telemetry, and a Chance to Make Things Better

Daniel "Spoons" Spoonhower, CTO and Co-founder



**Lightstep**

# Who am I?

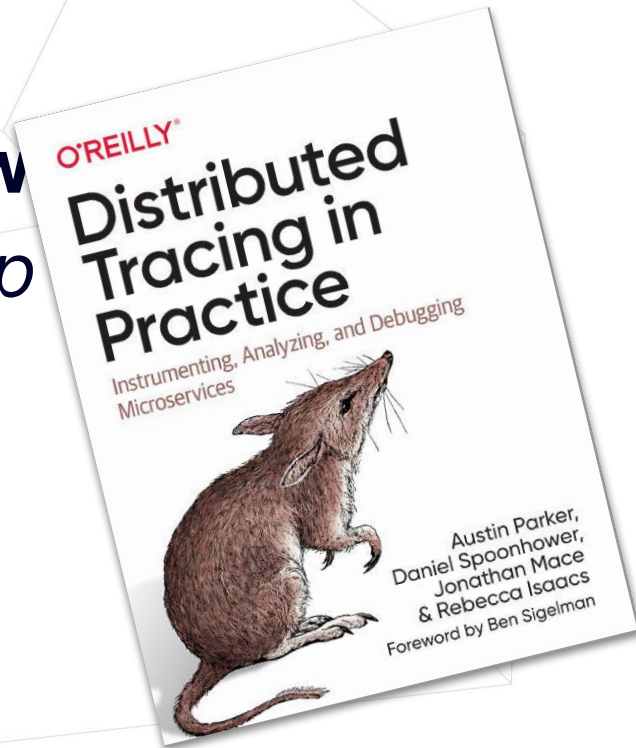
**Spoons (aka Daniel Spoonhower)**  
*CTO and Co-founder, Lightstep*



@save\_spoons

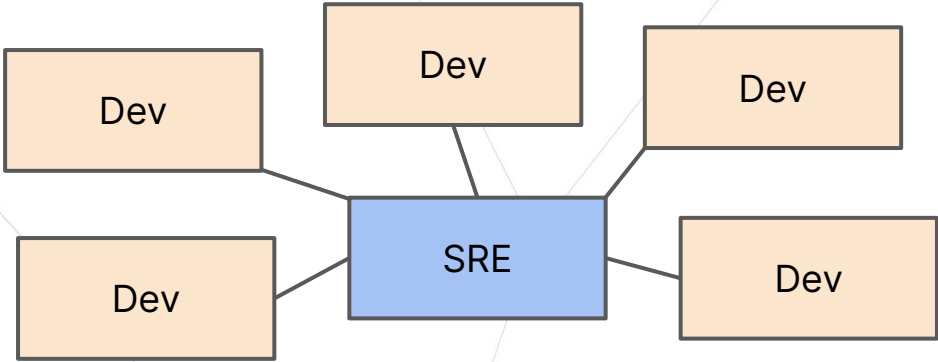
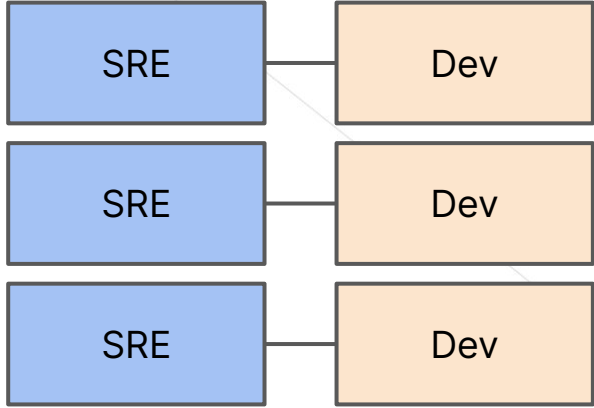
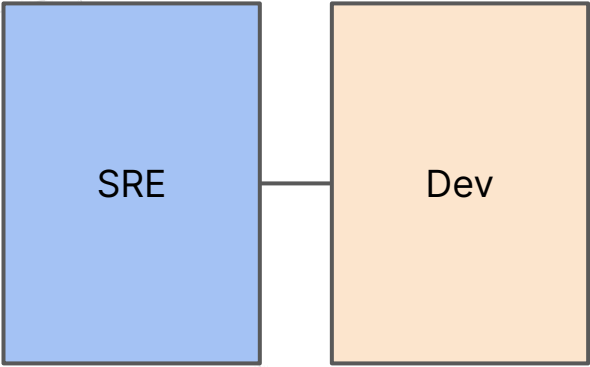


spoons@lightstep.com



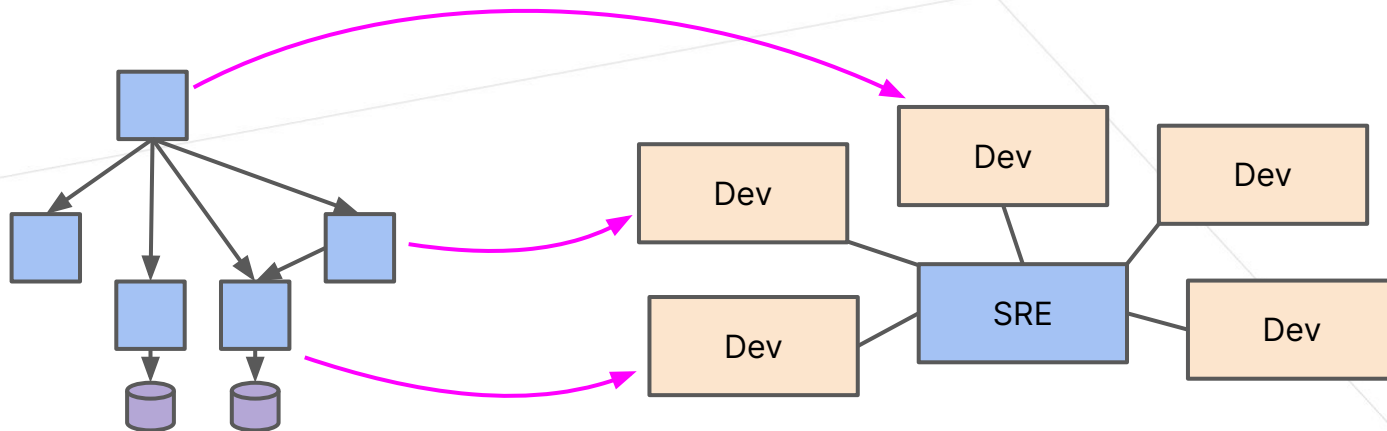
# Some Background







How do we create a map between our **software architecture** and our **organization**?



# Service ownership, defined

Dev teams are responsible for delivery of software *and* service

Includes activities such as:

- Writing code 😄
- Fixing bugs 😞
- Incident response
- Cost management
- Capacity planning
- ...

Increased independence

⇒ Higher developer velocity

Autonomous decisions

⇒ Better outcomes



# Obstacles to successful service ownership

You can't have independence without clear responsibilities and goals.

You can't scale autonomy without consistent ways of measuring and reporting on progress and outcomes.

You will meet (strong) resistance to changes ownership unless you also give teams the agency to change things.





Ownership means **Accountability** *and* **Agency**

Effective ownership requires **Distributed Tracing**

Importance of **Documentation**, **Oncall**, and **SLOs**



# Distributed Tracing





# Distributed tracing, defined

Traces are a form of telemetry based on *spans* with structure

- Span = timed event describing work done by a single service

Tracing is a diagnostic tool that reveals...

... how a **set of services** coordinate to handle individual user requests

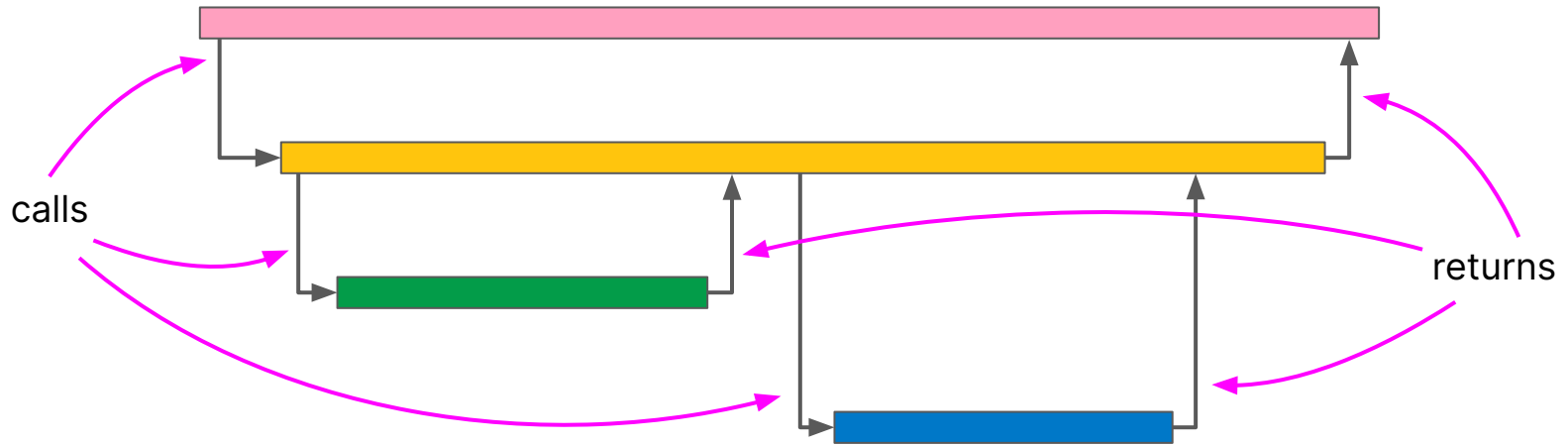
... from mobile or browser to backends to databases (**end-to-end**)

... including metadata like **events** (logs) and **annotations** (tags)

Provides a **request-centric** view of application performance



# Relationships matter



Traces encode causal relationships between callers and callees



Traces = raw material, not finished product

Dist

st structs

Dis

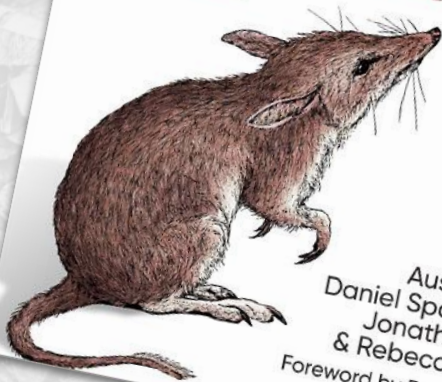
Google Technical Report dapper-2010-1, April 2010

**Dapper, a Large-Scale Distributed Systems Tracing Infrastructure**  
Benjamin H. Sigelman, Luiz André Barroso, Mike Burrows, Pat Stephenson,  
Manoj Plakal, Donald Beaver, Saul Jaspán, Chandan Shanbhag

O'REILLY

# Distributed Tracing in Practice

Instrumenting, Analyzing, and Debugging  
Microservices

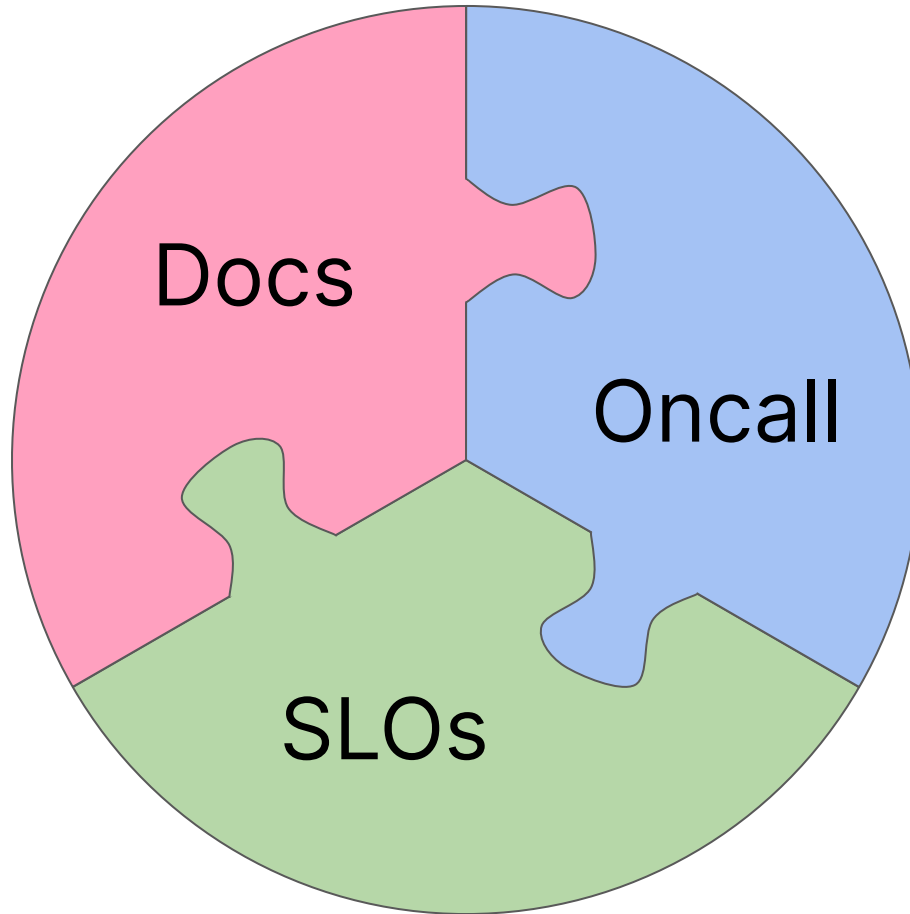


Austin Parker,  
Daniel Spoonhower,  
Jonathan Mace  
& Rebecca Isaacs  
Foreword by Ben Sigelman

  
OpenTelemetry

# Building Service Ownership







# Centralized documentation

LightStep / ... / Services

## Messenger

Name	Messenger
Description	Messenger is a gRPC service that provides an abstraction layer for delivering emails to Lightstep users.
Logs	<a href="#">Staging</a> <a href="#">Loadtest</a> <a href="#">Meta</a> <a href="#">Public</a>

**Purpose**  
Messenger is designed to provide an abstraction layer for sending messages to users outside of the Lightstep web application.

**Architecture**

```
graph LR; A[User Behavior Summary (cron job)] --- B[Insight Generators (Cron- or Service-Based)]; B --- C[Insights Notifier (cron job)];
```

Start with *expertise...* then *ownership*

Make it easy to find related:

- Telemetry and dashboards
- Alert definitions
- Playbooks

Use a template!

Track last-modified dates

- Require periodic audits & updates

# Centralized documentation

& Machine-readable

```
192 - name: kafka
193   team: data_ingest
194   properties:
195     build_type: helm
196     k8s_type: statefulset
197     skip_custom_values: true
198   environments:
199     staging:
```

Purpose  
Messenger is designed to provide an abstraction layer for the Lightstep web application.

Architecture

User Behavior Summary (cron job)

Insight Generators (Cron- or Service-Based)

Insights Notifier (cron job)

Make documentation *machine-readable*

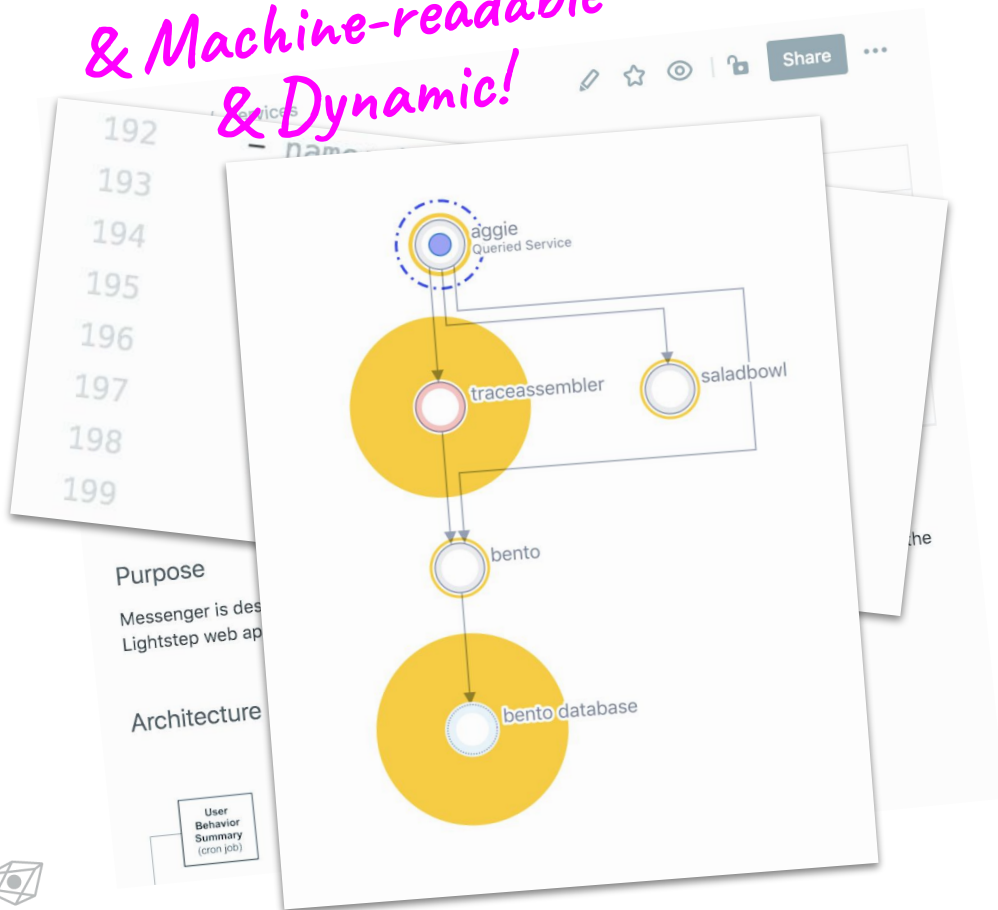
Use it to generate:

- Dashboard config
- Escalation policy config
- Deployment pipeline config
- ...

Make updating documentation *necessary* for day-to-day work

# Centralized documentation

& Machine-readable  
& Dynamic!



It's hard to keep service dependencies up to date *manually*...

**So don't!**

Use telemetry from the application

- Traces, in aggregate, reveal service dependencies
- Service levels show current reliability

# Why is documentation important?

Record who is accountable

Automate many mundane tasks

Train new team members

Build confidence



# Oncall

Oncall is (often) responsible for...

- Incident response
- Communicating status internally & externally
- Production change management
  - Deploying new code
  - Pushing infrastructure changes
- Monitoring dashboards
- Low-urgency alert triage
- Customer requests
  - And other interrupt driven work
- Shift handoffs
- Writing postmortems



Photo by [VT98Fan](#) and [Starwhooper](#)



# Iterating toward ownership

Establish a *need* to split

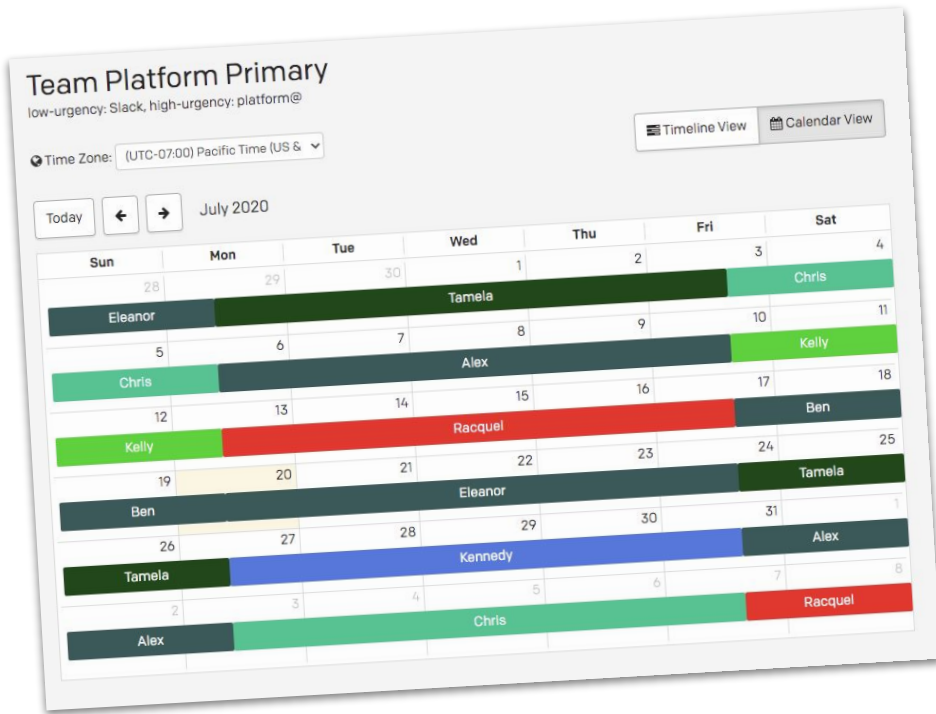
- Survey expertise & happiness
- Look at response time, number of people per incident

Some shock absorbers:

- Experts on the rotation
- Good documentation
- Balance between rotation size and number of services



# Handling alerts

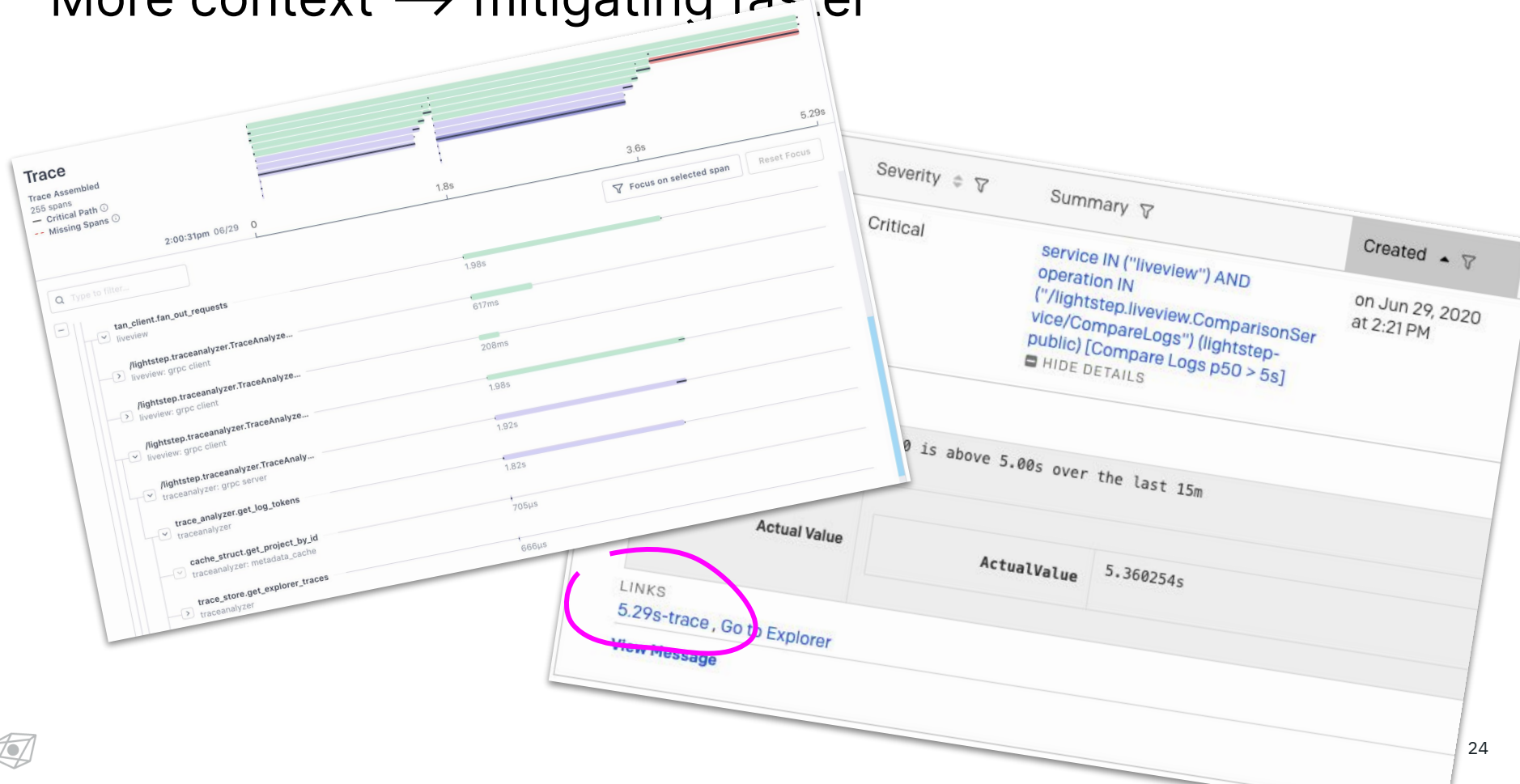


How to improve incident response:

- Reduce response times
- Deliver alerts to the right teams



# More context → mitigating faster



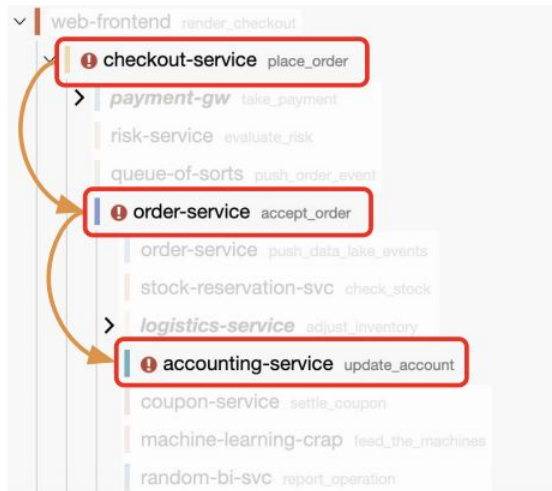


# Dynamic alert delivery

Send alerts directly to the teams that are responsible for taking action!

“Are We All on the Same Page?”  
Luis Mineiro @ SREcon19 EMEA

## WALKING THROUGH A TRACE



1. Starting at the span which was defined as the signal - **place\_order**
2. Inspect every child span's tags
3. Follow path with **error=true**
4. Rinse and repeat until no more children

# Handling alerts

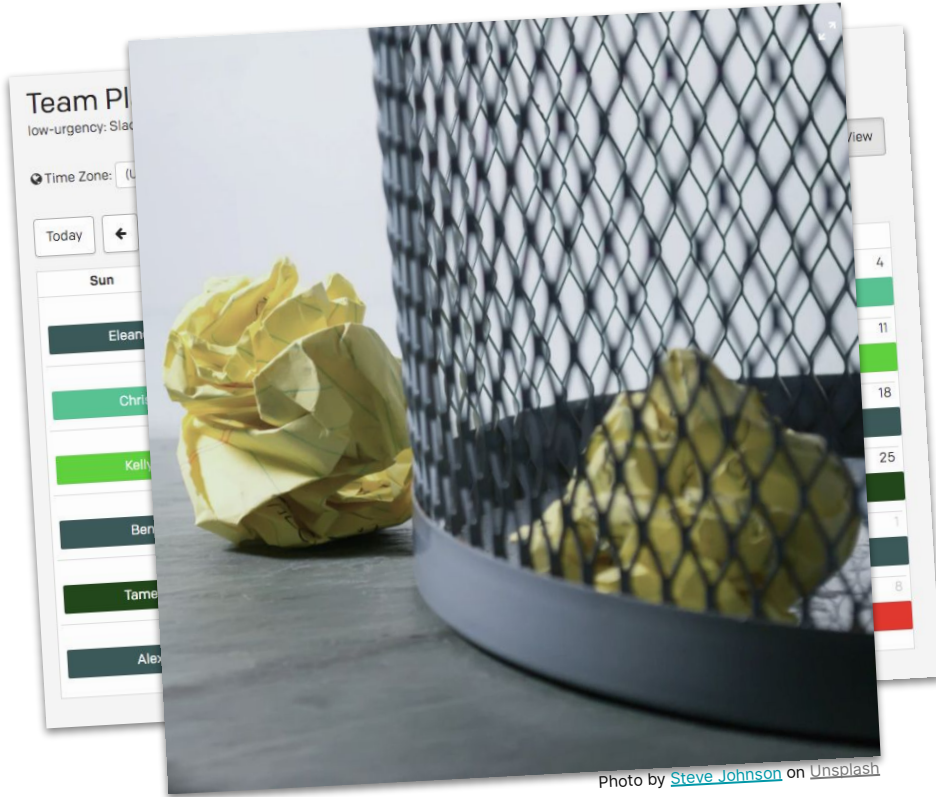


Photo by [Steve Johnson](#) on [Unsplash](#)

How to improve incident response:

- Reduce response times
- Deliver alerts to the right teams
- Delete unnecessary alerts
- Adjust rotation schedule to better fit team and sprint structure



# Improving postmortems

“How will we do better next time?”

- Ensure underlying issues are fixed
- Improve responses for novel issues

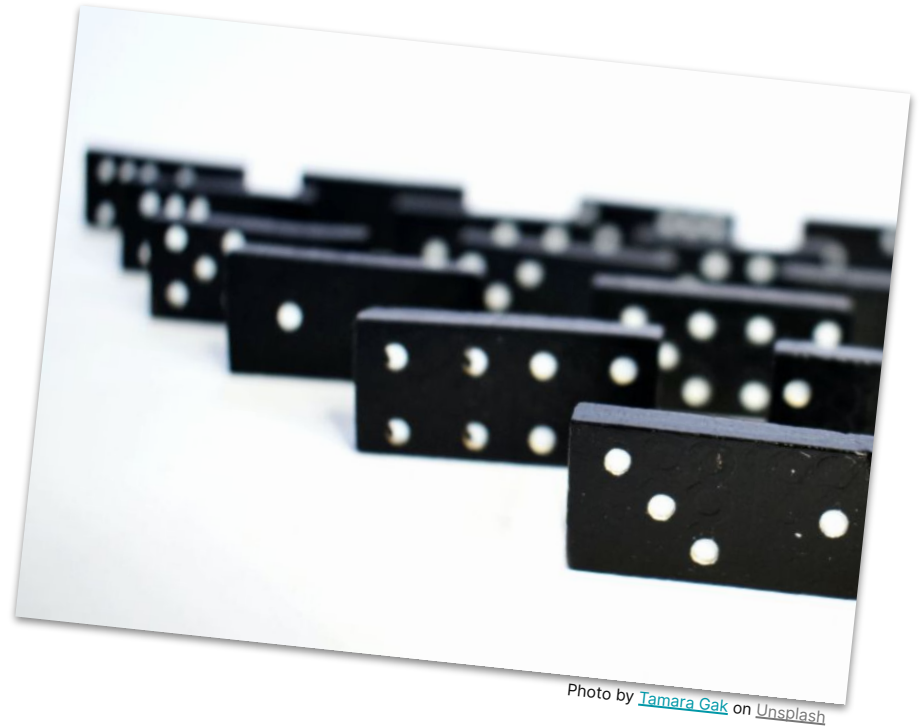


Photo by [Tamara Gak](#) on [Unsplash](#)



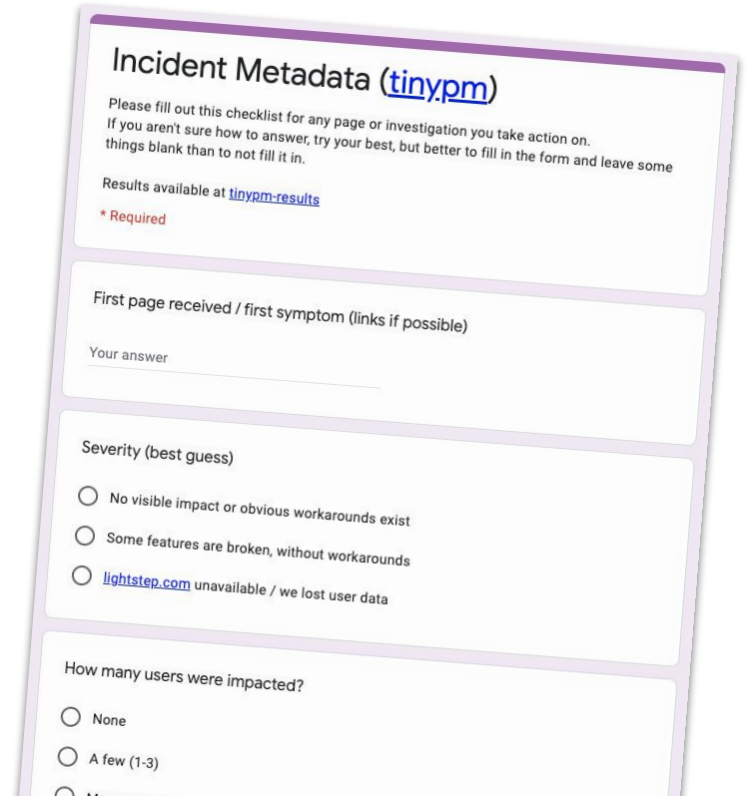
# Postmortems are *documentation*

Make it easy!

Make sure they are centralized

Make sure action items and other info are captured in a structured way

(And of course) leverage telemetry



**Incident Metadata ([tinypm](#))**

Please fill out this checklist for any page or investigation you take action on.  
If you aren't sure how to answer, try your best, but better to fill in the form and leave some things blank than to not fill it in.

Results available at [tinypm-results](#)

\* Required

First page received / first symptom (links if possible)

Your answer \_\_\_\_\_

Severity (best guess)

- No visible impact or obvious workarounds exist
- Some features are broken, without workarounds
- [lightstep.com](#) unavailable / we lost user data

How many users were impacted?

- None
- A few (1-3)
- Many



# Why is improving oncall important?

Direct impact on customer experience (revenue, reputation, etc.)

Time spent handing pages, writing postmortems, handling interrupts is...  
time **not** spent building new features, proactive optimization

Stress of oncall has major impact on job satisfaction





# Determining SLOs

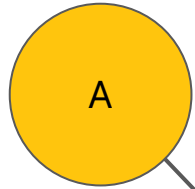
Ask:

- What do your customers expect?
- What can you provide today?
- How do you expect that to change?

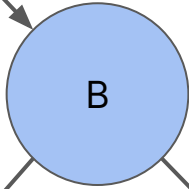


# Derive internal SLOs using tracing

A: p99 latency < 5s

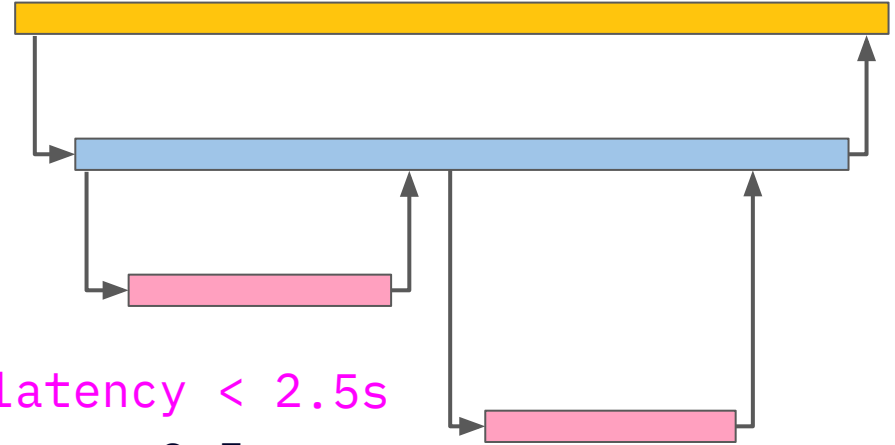
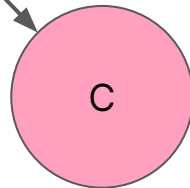
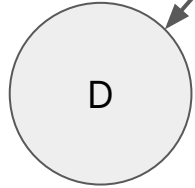


B: p99 latency < 5s



~p99.5 latency < 2.5s

C: p99 latency < 2.5s





# Why are SLOs important?

They measure success in delivering service

Teams use them as a guide to prioritize work

Consistency and transparency across your org

- Hold teams accountable in a uniform way



# 3-piece puzzle review

## Documentation

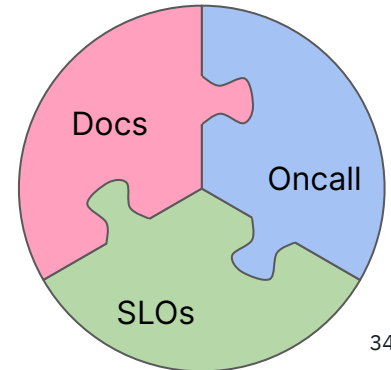
- Establishes ownership: who you will hold accountable
- Don't try to manually manage dependency lists, etc.

## Oncall

- More than just incident response
- Use docs and telemetry as part of investigation, automation, communication

## SLOs

- How you communicate and measure success!
- Define objectives for internal services using tracing



Next steps...



# Making changes

Rolling out new processes/tools with many teams is hard

1. Process/tools must provide **value to dev teams**
2. Ideally, they are **necessary** parts of their day-to-day work

To establish and maintain service ownership

- Use a combination of docs, oncall process, and SLOs
- *Manufacture* a need for those process/tools where necessary
- Give teams a budget for improving docs, alerts, and reliability



# Ownership = Accountability + Agency

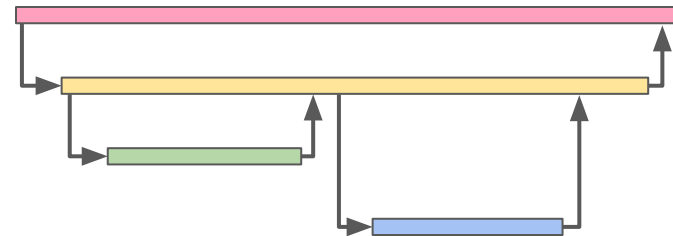
## Accountability

- Set deliverables and goals for service owners
- Judge their performance based on those deliverables and goals

## Agency

- Offer the information, confidence, and budget to improve

Telemetry provides key information to drive both!



# Thank you

 @save\_spoons

Daniel "Spoons" Spoonhower, CTO and Co-founder

