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

Daniel “Spoons” Spoonhower, CTO and Co-founder
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

Distributed traces – basically just structs

Distributed tracing – the art and science of deriving value from traces

OpenTelemetry
Building Service Ownership
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

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
Handling alerts

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
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
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
SLOs

Service Level Objectives
- Customer expectations for the service your provide
  - Both external and internal customers
- With a threshold that lets you report success/failure over time (or groups of events)
- As a means of communicating how reliable the service is

Aim for less-than-perfect
- three nines, 99.5%, 98.2%, ...

Common indicators
- Latency (p50, p99, etc.)
- Error rate

But specific to the endpoints, operations, and flows of your application

**p99 latency < 5s over the last 5 minutes**

service level indicator (SLI)

threshold

measurement window
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

C: p99 latency < 2.5s

~p99.5 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