Measure What Matters

Observability Day
NORTH AMERICA

November 12, 2024
Salt Lake City

Jamie Danielson
Software Engineer at Honeycomb
Maintainer on OpenTelemetry JS

KubeCon | CloudNativeCon

North America 2024

Measure What Matters

Jamie Danielson, Honeycomb

Measure What Matters

Jamie Danielson

Software Engineer at Honeycomb
Maintainer on OpenTelemetry JS

Measure What Matters

Agenda

- Reduce the noise
- Customize instrumentation
- Set SLOs for actionable insights

Reduce the noise

Reduce the noise

Nuisance alerts

- Alert: High CPU Usage
- Alert: … other team’s service

If you wait until a customer reports a problem before acting… why have these alerts at all?

Reduce the noise

Cost of alert fatigue

- Distraction, lowered team morale, increased stress
- You may neglect to investigate an actual failure scenario because you're ignoring the signs

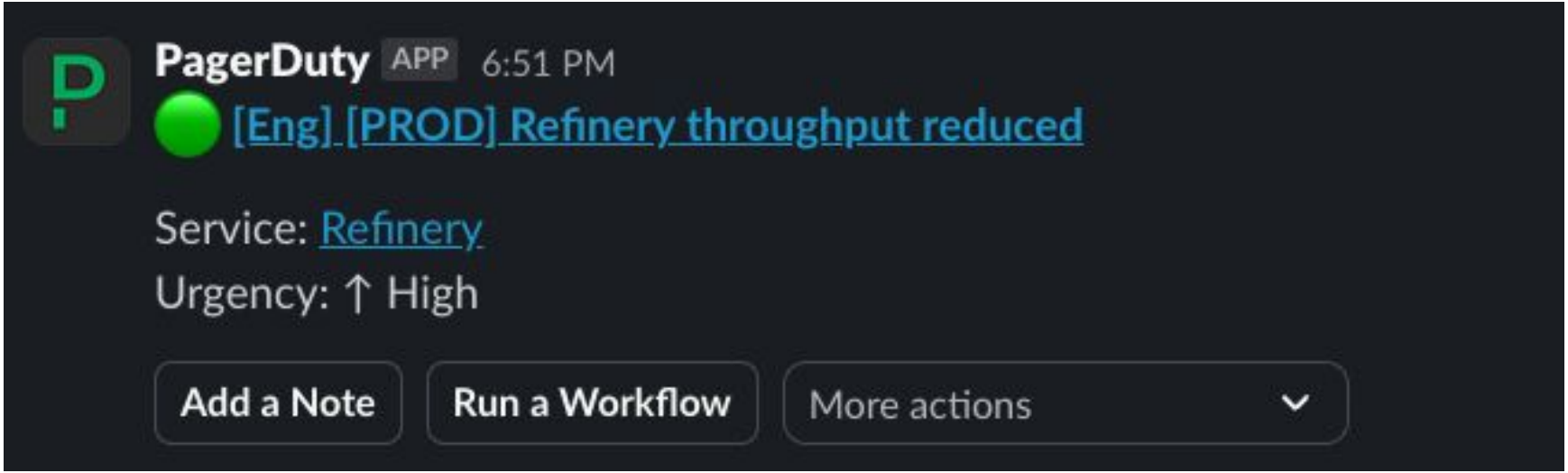
Reduce the noise

"...instead of counting outages and alerts, we should focus on *whether we react to them in a useful manner*"

- Fred Hebert, Tracking on-call health

Reduce the noise

Prioritize actionable alerts



Reduce the noise

Prioritize actionable alerts

- If the current value being measured does not give enough useful information to start investigating… find another value.

Reduce the noise

Prioritize actionable alerts

“Failures that get automatically remediated *should not trigger alarms*.”

- Liz Fong-Jones, Observability Engineering

In the case of a self-healing system, failures automatically remediated shouldn’t trigger alarms but instead create tickets for business-hour review.

Customize instrumentation

Customize instrumentation

Default metrics aren’t enough

- Averages can be misleading
- Spikes in resources may not represent user impact
- Simple metrics do not provide enough information to debug

Customize instrumentation

50th, 85th, 95th, and 99th percentile latencies for a system.
From [Google SRE book]

Customize instrumentation

OpenTelemetry

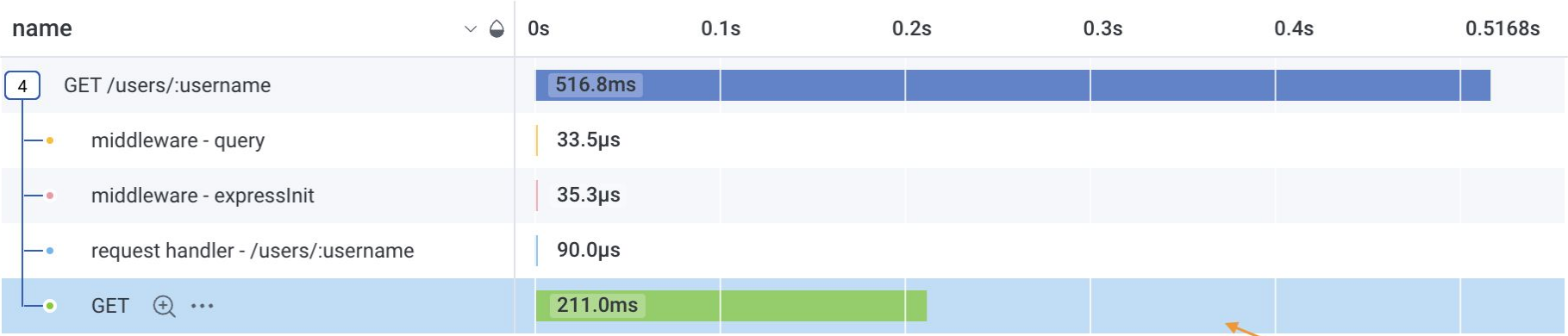
OpenTelemetry is a standardized way to instrument, generate, collect, and export telemetry data.

Customize instrumentation

Auto-instrumentation with tracing can get you started

- You can get tracing from Service A to Service B
- But you need to trace further logic within Service B if it turns out to be too high level

Customize instrumentation

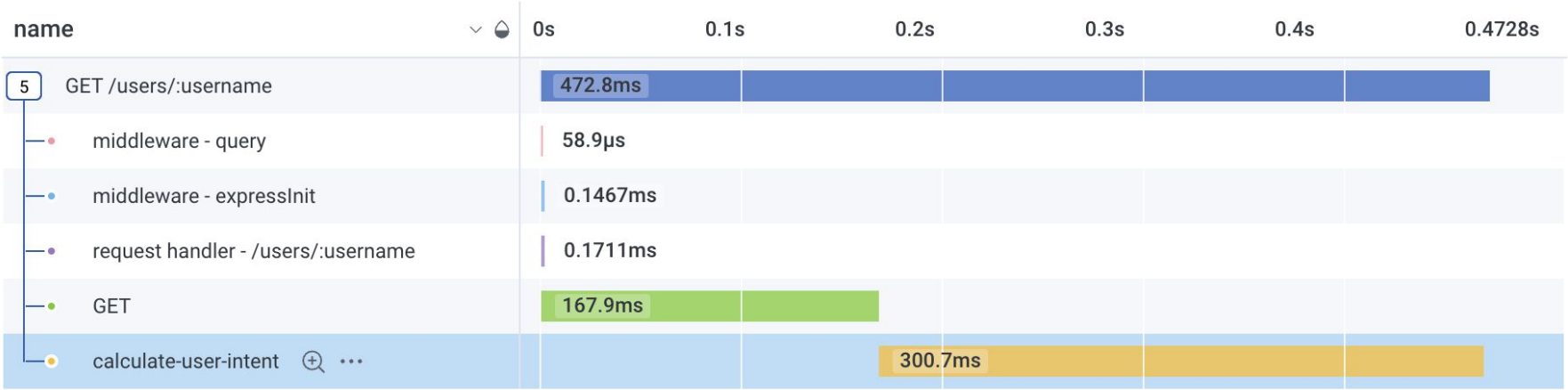


What happened in here?

Customize instrumentation


```
const getUserHistory = async (username: string) => {
  const id = getUserId(username);
  const response: UserInfo = JSON.parse(
    (await makeRequest(`${userServiceUrl}/${id}`)) as string,
  );
  trace.getActiveSpan().setAttribute(USERID_ATTRIBUTE, response.id);
  await tracer.startActiveSpan('calculate-user-intent', async (span) => {
    await calculateUserIntent();
    span.end();
  });
  return id;
};
```

Customize instrumentation



Customize instrumentation

Context-specific attributes

- Geographical information
- Logged in vs anonymous users
- Device type and screen size
- App version in use
- Feature flag enabled
- Discount applied
- …anything else that might be useful

You don’t know what will be different the next time a failure mode occurs.

Customize instrumentation

Context-specific attributes about the state of the app

- App version in use
- Feature flag enabled
- Discount applied

You don’t know what will be different the next time a failure mode occurs.

Set up SLOs

Set Service Level Objectives (SLOs)

What is an SLO?

- SLO: Service Level Objective
  - a goal for measuring service health, or a reliability target
- SLI: Service Level Indicator
  - a specific measurement to define service health

Example SLI: Checkout page loads in 2 seconds

Example SLO: That SLI is true 99.5% of the time

Set Service Level Objectives (SLOs)

Setting realistic, impactful SLOs

- Do not aim for perfect. 100% is not admirable; it is unrealistic.
- Tie SLOs back to business impact
  - Different SLOs may be needed for different scenarios
- For non-critical signals, create a task to investigate during business hours
  - If user impact is minimal or non-existent, do not page.

Set Service Level Objectives (SLOs)

“The goal of an SLO is to provide a useful signal: if including a specific datapoint dilutes the signal, then it may be worth excluding them or creating a different measure for them.”

- Quail, SRE at Honeycomb

Set Service Level Objectives (SLOs)

Setting realistic, impactful SLOs

Remember that Refinery alert I talked about earlier?

We realized we had another attribute that more reliably indicated a problem with the service itself, called stress_level.

We now use that in an SLO, and only page if there was a sharp increase in stress_level over a short period of time.

Set Service Level Objectives (SLOs)

Revisit and refine over time

- ~~Happiness~~ Observability is a journey, not a destination.
- Newer business requirements may mean changes to SLOs
- Check in with your team and run retros on recent incidents.

Set Service Level Objectives (SLOs)

"**Things we find and hear nothing about** means we may be sensitive enough (or maybe over-sensitive); **things we find and hear about** means we're seeing their pain (and maybe not seeing it early enough); **things we don't find but customers do** are a sign we may have under-sensitive signals."

- Fred Hebert, Alerts are Fundamentally Messy

Find problems before your customers do.

Measure What Matters

Recap

- Reduce the noise
- Customize instrumentation
- Set SLOs for actionable insights

Measure What Matters

Resources

- Observability Engineering
- Google SRE Book
- Tracking on-call health
- Alerts are fundamentally messy

feedback, slides (soon)

Thank you!

Jamie Danielson, Honeycomb

Say hi to our team in the lobby and pick up some stickers and pins!