Outreach Engages Their Production Code With Honeycomb

ABOUT

Outreach is the number one sales engagement platform with the largest customer base and industry-leading usage. Outreach helps companies dramatically increase productivity and drive smarter, more insightful engagement with their customers.

ENVIRONMENT

Ruby, Go, NodeJS
MySQL, Kafka
Kubernetes containerized on AWS
Cloudwatch
Datadog for log management
Wavefront for metrics

GOALS

In 2015, the Outreach engineering team relied on querying logs with Elasticsearch to pinpoint production issues. By 2019, booming customer adoption made that approach unmanageable.

Initially, searching through logs for general criteria, like group IDs, let them understand which customers might be experiencing issues. At scale, their approach of creating one log entry per HTTP request resulted in millions upon millions of log entries that rendered their previous search methods unusable, even with a two week retention period. The team wanted a better approach than millions of log entries and limited view metrics.

WHAT THEY NEEDED:

The team considered a metrics-based approach for discovering production issues. But a key requirement was pinpointing performance for specific users and groups, rather than analyzing performance in aggregate. They needed to quickly sift through high-cardinality data to find performance bottlenecks.

Amritansh Raghav, VP of Product & Engineering, joined Outreach in mid-2019 and recommended a new approach: proactively discover production issues with observability and take a closer look at Honeycomb.

Primary Use-Case: Performance Optimization

The Outreach team eventually settled on Wavefront for general-use metrics. To pinpoint bottlenecks, they explored distributed tracing with Jaeger. Trace views provided incredibly useful insights, but the team found the Jaeger user experience challenging. After a side-by-side evaluation between Jaeger and Honeycomb, they chose Honeycomb because its UX was better, more intuitive, and easier to adopt.

The team monitors their AWS RDS database performance by exporting slow queries with durations over a specified threshold to Cloudwatch. The team's natural inclination was to investigate slow queries via Cloudwatch. But Cloudwatch logs only identify the slow SQL query and don't provide any context around why latency might be occurring. It's easy to see a slow query is occuring, but not why it is slow or who that slowness is affecting.

Similarly, Wavefront may detect request latency on an API endpoint and trigger an alert. But the team doesn't rely on Wavefront metrics to diagnose issues. Instead, they turn to Honeycomb to examine traces for the affected API endpoint. They group-by relevant dimensions to understand where bottlenecks are occurring and introspect from there. Now, the Outreach team uses Honeycomb for continuous performance tuning.

"Honeycomb has definitely helped us when the thing that's going wrong is new. I like not being blind anymore. With Honeycomb we're generally not running around wondering where the problem is. We can always figure out what component is causing the problem."
–Richard Laroque, Software Engineer at Outreach

The Bridge Into Observability

The Outreach SRE team predominantly focuses on infrastructure rather than writing application code. But Outreach creates a culture of service ownership by ensuring all engineers are in on-call rotations with escalation policies; a key success factor in their observability journey. Many engineers learned to use Honeycomb during internal lunch-and-learn sessions.

"I'm a big Honeycomb fan and I think everyone in engineering should use the product. Early on, after an incident, I wrote a step-by-step guide that really helped the team understand how to better use the tool." **–Richard Laroque**

Today, some Outreach engineers are well-versed in using observability to approach any problem as new. They leave behind prior assumptions and use Honeycomb to guide their investigations, asking questions, whose answers reveal the next question, until they find the right bottleneck.

"Once you become familiar with how to find the trace parent ID, then isolate trace spans from leaf-level spans, you are on your way to locating an issue or root cause." **–Richard Laroque**

But not all engineers know where to start. To help the team learn from one another, Outreach uses Honeycomb boards to provide useful starting points for investigation.

"A handful of boards have been set up to help new team-members navigate to existing queries and use that as a jumping off point. If someone is new to an on-call rotation, debugging can take a little longer especially if it's a Monday morning traffic surge. Our DevOps channel is filled with links to Honeycomb queries and teams are picking it up pretty quickly." **–Richard Laroque**

Beginning to Implement SLOs

Outreach has started using Honeycomb's Service Level Objectives (SLO) feature. They're taking time to thoughtfully consider the implication of measuring Service Level Indicators (SLIs) across time frames wider than what just happened in the last 10 minutes.

"Our SLOs are now hooked up to a Slack channel and are working just fine. Honeycomb got SLOs right. You have followed the practices in the Google SRE book really well. I haven't seen anyone else in the market doing this the right way." **–Richard Laroque**