honeycomb.io

Building Your Observability Practice with Tools that Co-exist

> In the world of metrics, monitoring, APM, tracing, and related tools for debugging and troubleshooting, many vendors focus on what makes their product different and better, and why it is the only thing a customer should use. We've certainly done some of that--we believe that Honeycomb is by far the best tool on the market for digging into gnarly bugs that require the ability to query and break down by the most granular data field (aka high-cardinality values)--but what we sometimes fail to point out is that Honeycomb is not just for when all your other tools have failed you.

Not only that, but Honeycomb is perfect to use alongside the things you're already using to develop, deliver, and operate your service. If you've been in business for any amount of time, you've already made choices--choices about what data you're collecting, how you're collecting it, where you're sending it, how you're visualizing it--and you've no doubt put work into making these tools serve your needs. And although Honeycomb can achieve a lot of what you are doing with the tools you've got in-house already, you don't need to use every feature and capability of Honeycomb to get your money's worth.

To put it another way, the goal is to augment and expand your ability to deliver your service and improve your bottom line, not require you to rip out the tooling your organization has depended on up to now. You can benefit from using Honeycomb alongside what’s been working for you until now.

The reality is that team, skills, and cultural transformation for DevOps is hard and takes time. As your organization evolves toward service ownership and a practice of observability-driven development, you’ll require a different set of tools and processes, and this kind of transformation doesn’t occur between one deploy and the next. While you develop and improve your observability practice, Honeycomb coexists as part of the ecosystem of tools that bring value to your business.

The things everyone knows they need

Most organizations share the following basic requirements:

Some metrics, to keep a high-level eye on infrastructure

Metrics tools do well at counting things like number of jobs queued, host level resource usage, number of requests served by the system, and so on. Your organization has likely built many dashboards to monitor your service, and tuned them over the months and years to reflect the particular foibles of your production environment and user base. There may be legacy documentation in place instructing engineers to optimize their logging for output to Graphite, or Librato. These systems aren't going to be dismantled overnight, nor should they be.

An archive, for longer-term storage and compliance

Many business practices have legal requirements for data retention timelines, and many executives are simply more comfortable in the knowledge that your organization has a searchable archive going back six months, a year, or even several years. For the majority of forensic or compliance use cases, ETL and query response timelines don't need to be fast, as long as you can provably retrieve the data as required by your business. Perhaps you built out an ELK cluster for this purpose. It might even have originally been your team's primary troubleshooting interface but was relegated to the role of archive when scaling it to respond quickly during a production incident no longer proved cost-effective.

A query and graphing tool, to investigate and debug

Here’s where you’re likely to be seeing the most obvious need for Honeycomb to augment your existing tooling. If you’ve been using classic APM like Datadog or New Relic for a while, you might have found their approach met your needs at the outset, but as your environment and use case has become broader and more complex, you may be experiencing slower performance and costs spike as you

add more tags, or noticing that more and more issues are not solvable using the preset views available.

Your tools should coexist while your enterprise scales

When Honeycomb coexists with classic log-aggregation and metrics tooling, most organizations start out by using an integration to send existing log output to Honeycomb, sometimes also duplicating that stream and sending it to their existing archive and/or metrics tools. They typically also install one or more Honeycomb Beelines, which automatically instrument their code with events and traces and send the telemetry as well.

In this situation, when investigating something turned up by metrics, an oncall engineer may begin their troubleshooting by turning to what's familiar--Datadog or New Relic, Splunk, or another classic APM tool. When an issue's complexity outstrips the abilities of these tools, and the services involved are sending data to Honeycomb, the engineer is able to drill in more deeply and gain resolution.

> **Geckoboard** built their own system health dashboards using Librato to display high-level throughput, latency, and error rate-type data, and have been using standard log search tools since the time when log search was new and shiny. But as their business has scaled up, so has their need to understand specifically who is affected by a given issue—and although the log search bill was growing, Geckoboard's increasing requirement for breaking down by things like customer ID still wasn't being met. Find out how they use Honeycomb alongside their legacy log search tools to keep up with their growing business needs:
>
> Geckoboard Gets the Context They Need to Keep Customers Happy Without Breaking the Bank

In the past, **LaunchDarkly** had success using an in-house Graphite instance to collect the metrics they used to monitor customer health, but as their business grew, they found it could not keep up with the increasing volume and cardinality of the data stream. They would know something was wrong, but could not get the performance or detail they needed to find out what it was or who it was affecting.

Graphite still met their needs for monitoring some aspects of system health, but they needed something better able to dig into how their users actually experience their service. Read more about how LaunchDarkly's team uses Honeycomb to give their dashboards new life and continue to deliver the quality of service their customers depend on:

[LaunchDarkly More Confident with Next-Gen APM]

Incident Response. Ongoing Optimization. Ongoing Development

At this stage, the development organization is beginning to see the first wave of benefits from the observability that Honeycomb offers. The team has solved the first few previously-unsolvable issues, squashed some performance-affecting bugs, and the quality and stability of application code is on an upward trajectory. What's next?

Observability-Driven Development is an ongoing journey

It's important to remember that observability has a definition: **the ability to ask and answer the questions about your systems that you need to without shipping new code**. This matters, because logs, traditional APM, and monitoring tools all claim to give you observability today, but ultimately don't give you the ability to do that in every case, or without vastly exceeding your budget.

As legacy services are replaced with new code, the engineering team can take the opportunity to instrument with observability in mind, implement additional Beelines, and improve the overall telemetry sent to Honeycomb. And when they do this, much more is possible; the entire organization also benefits from more comprehensive performance analysis tooling, greater visibility into third-party services that impact your bottom line, and greater data-driven insights when prioritizing both product and technical debt work.

Ongoing Performance Analysis

When a new feature is shipped, Honeycomb can show you the impact it has on your system. Observability means you can more easily prioritize where to add capacity or optimize code, where to focus in order to make the most impact and keep your important customers happy.

Intercom used Honeycomb to evaluate performance across all the dimensions required to understand how different users and types of usage affected the performance of a given endpoint. They were able to both identify the portions of the code needing refactoring as well as document concrete examples of how they'd improve performance.

How Intercom sped up their busiest endpoint (by as much as 50%)

Addressing Technical Debt

As your organization scales and your product's footprint grows, you must maintain clear sight-lines across your infrastructure. Distributed tracing allows engineers to evaluate systems performance so they can better understand the interactions among an increasing number of services and make informed decisions about when to burn down technical debt vs chasing the latest shiny technology.

carwow leveraged Honeycomb to follow a request through its entire life-cycle and understand the impact on different subsystems in the code, leveraging its cross-team collaboration features to solve issues:

Honeycomb Tracing Drives Efficiencies as carwow Scales

User Happiness and Product Management

Modern services have moved beyond simple high-level metrics. Understanding how the end user experiences your product is critical to the success of your business. Observability means you have the power to notice when they use features in unexpected ways and can capture that data for your product team to investigate and leverage.

Intercom discovered one of their users was trying so hard to use their product in ways they hadn't anticipated that it was impacting the overall experience of many others—and as a result informed future product planning for that feature:

Intercom <3 Honeycomb

Check any that apply

If you're reading this, you are already somewhere on the path to needing and implementing observability tools for your service. How close are you? Some contributing factors:

- ☐ You're moving to a microservices architecture
- ☐ You have a large-scale monolith application with stringent SLOs and a high code ship velocity
- ☐ Your organization has a mandate to move toward code ownership by engineering
- ☐ Your service depends on a growing number of third-party APIs
- ☐ Oncall morale is trending lower
- ☐ The rate of unsolved incidents is growing

If you checked one or more of these boxes and you aren't already using Honeycomb, your current tooling is probably either costing you too much or not fully meeting your needs (or both).

Have it your way while you find your way

Tool overload is never a good thing. Having said that, if your team has acquired muscle memory for and is still getting value out of an existing tool, it could be too disruptive to abruptly switch their tooling and processes for a new approach.

Many organizations have discovered that running Honeycomb alongside legacy tools makes it possible for engineers to find and solve the issues those tools can't, while at the same time iteratively improve the organization's code quality, level of technical debt, customer happiness, and much more via Observability-Driven Development---all without breaking the bank.

About Honeycomb

Honeycomb provides next-gen APM for modern dev teams to better understand and debug production systems. With Honeycomb teams achieve system observability and find unknown problems in a fraction of the time it takes other approaches and tools. More time is spent innovating and life on-call doesn't suck. Developers love it, operators rely on it and the business can’t live without it.

Follow Honeycomb on Twitter LinkedIn
Visit us at Honeycomb.io