Buyer's (and Evaluator's) Guide to Observability

honeycomb.io

Overview

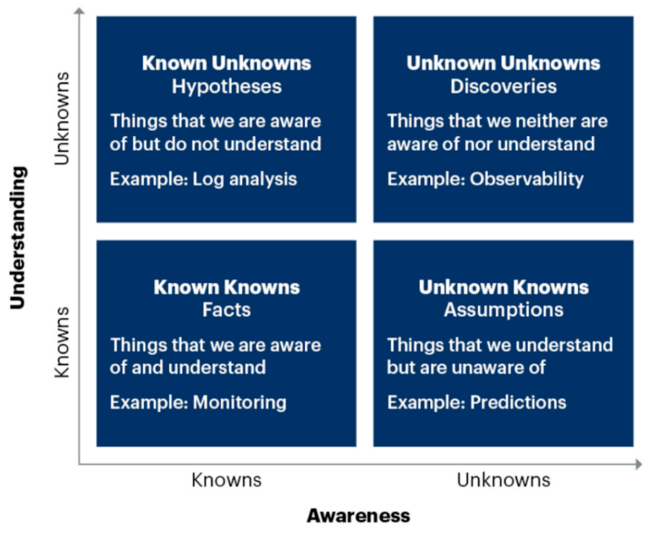
Critical features & capabilities to look for in a dedicated, purpose-built tool.

Evaluating and selecting an observability tool can be daunting. You need to approach evaluation with a solid understanding of the outcomes you ultimately want to achieve. When vendors tell you it's possible to ask any new question of your system and get fast, reliable answers, how do you know they're going to deliver on that promise? Different tools are designed for different purposes across the software engineering lifecycle.

Perhaps you are actively using a monitoring tool if you are dealing with any level of system complexity. You may have invested in a logging or APM tool to proactively run queries to better understand availability, performance, and rate of errors. Log management tools give you the ability to search when something needs deeper investigation but you really have to know what you are looking for. If you are down the path of understanding how end-users interact with your code as you scale and ship updates, you may be off to a great start. Are you handling incidents in the most efficient way or do you burn cycles figuring out the source of problems with time to resolution taking longer than you'd like?

According to a June 2020 Gartner report titled "*Monitoring and Observability for Modern Services and Infrastructure*," monitoring by its very definition suggests that you know what you are looking for, which is often referred to as 'known knowns' and 'known unknowns'. But what happens when you don't know what you are looking for? For modern distributed systems managed by multiple teams, it is impossible to understand the details of an application's internal state, not to mention whether it's performing well for everyone. In this same report, Gartner lays this out in an easy to read quadrant view.

Quadrants of Awareness and Understanding



Source: Gartner Monitoring and Observability For Modern Services and Infrastructure report by Gregg Siegfried, June 02, 2020 (ID: G00720854)

In order to discover unknown unknowns, you need some internal system expertise or instrumented code that provides insightful telemetry so you (or anyone on the team) can perform deep exploratory analysis.

Honeycomb has worked with engineering teams of varying sizes and maturity and we have learned that the most commonly asked for features include those listed in the table below. Listening to customer feedback has not only shown us why they choose us, but also reveals what's most important to them. The table below (figure 1) shows what types of features and capabilities are the most critical when it comes to awareness and understanding of why services are behaving in a certain way.

The six key requirements to consider as you move forward with evaluating an observability tool are detailed in this document which will hopefully guide you when considering a purpose-built observability tool. Honeycomb scores well compared to APM tools (New Relic and DataDog) and Log Management (Splunk) and one other Observability tool (Lightstep).

	Honeycomb	New Relic	Lightstep	Datadog	Splunk
01. Well-integrated tracing	✓	~	✓	~	✗
02. Anomaly explanation with actionable insights	✓	~	✓	~	✗
03. High-resolution data out of the box	✓	✗	✓	✗	✗
04. Sophisticated engineering team ergonomics	✓	✗	✗	✓	✗
05. Rapid SLO iteration	✓	✗	✗	✗	✗
06. Actionable insights for error budget burn	✓	✗	✗	✗	✗

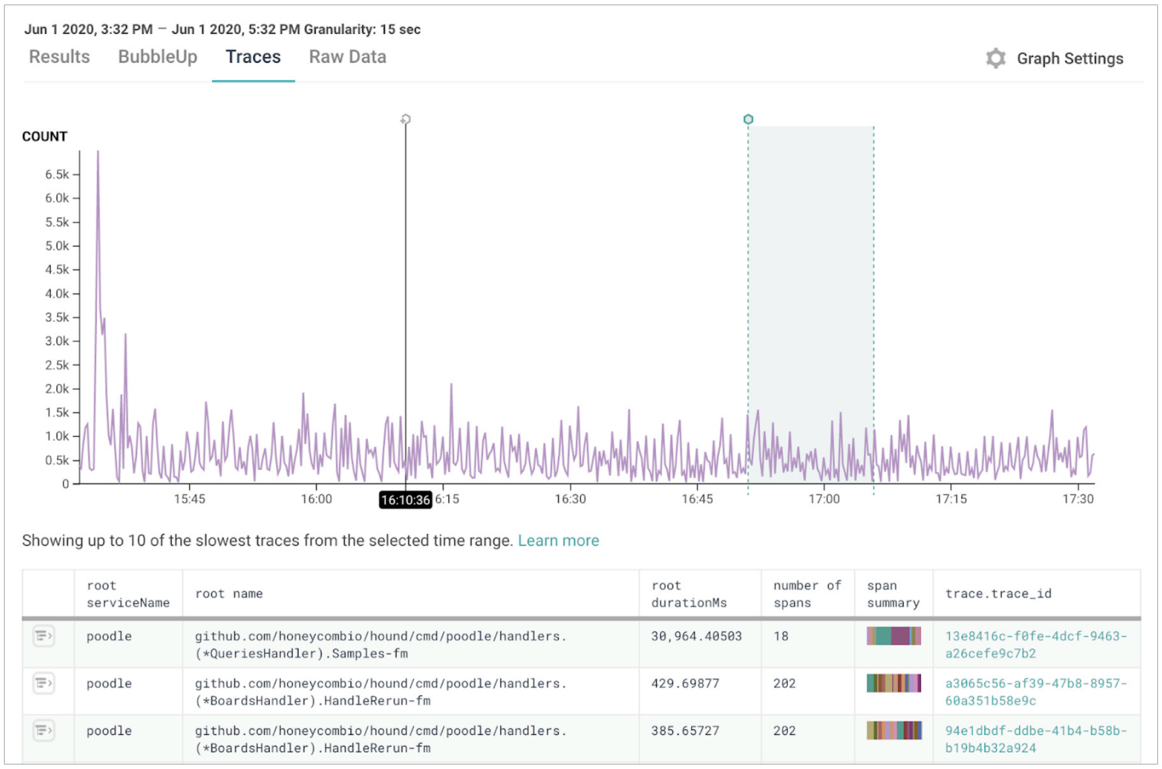
✓ Excellent Support ~ Moderate Support ✗ Little or No Support

Figure 1: Answers to the most frequently asked for feature & capability set in an observability tool evaluation.

01
Well-integrated tracing

Does the system have tracing as a core product feature?

Many tools in the market today have some support for a distributed tracing view of your systems, which shows where latency is happening as a request spans multiple services or through distinct chunks of work within one service. Unfortunately, tracing for many vendors is an add-on or afterthought, and not part of the core product. This can lead to a poor experience because the product wasn't built with tracing in mind, and awkward relationships with vendors as they treat tracing as an upsell rather than something provided out of the box. With Honeycomb, tracing comes standard, and toggling back and forth between a distributed trace view and other views of the system such as a heatmap or histogram is seamless. With no need to context switch or open up an entirely new application, you find answers fast and validate what you are seeing with reliable data-centric views.



Showing up to 10 of the slowest traces from the selected time range. Learn more

	root serviceName	root name	root durationMs	number of spans	span summary	trace.trace_id
	poodle	github.com/honeycombio/hound/cmd/poodle/handlers.(*QueriesHandler).Samples-fm	30,964.40503	18		13e8416c-f0fe-4dcf-9463-a26cefe9c7b2
	poodle	github.com/honeycombio/hound/cmd/poodle/handlers.(*BoardsHandler).HandleRerun-fm	429.69877	202		a3065c56-af39-47b8-8957-60a351b58e9c
	poodle	github.com/honeycombio/hound/cmd/poodle/handlers.(*BoardsHandler).HandleRerun-fm	385.65727	202		94e1dbdf-ddbe-41b4-b58b-b19b4b32a924

Figure 2: Result set selects a specific time range to show the slowest traces or service requests which you can further drill into to learn more details.

02
Anomaly explanation with actionable insights

Can the system quickly identify which traces are of interest to troubleshoot production?

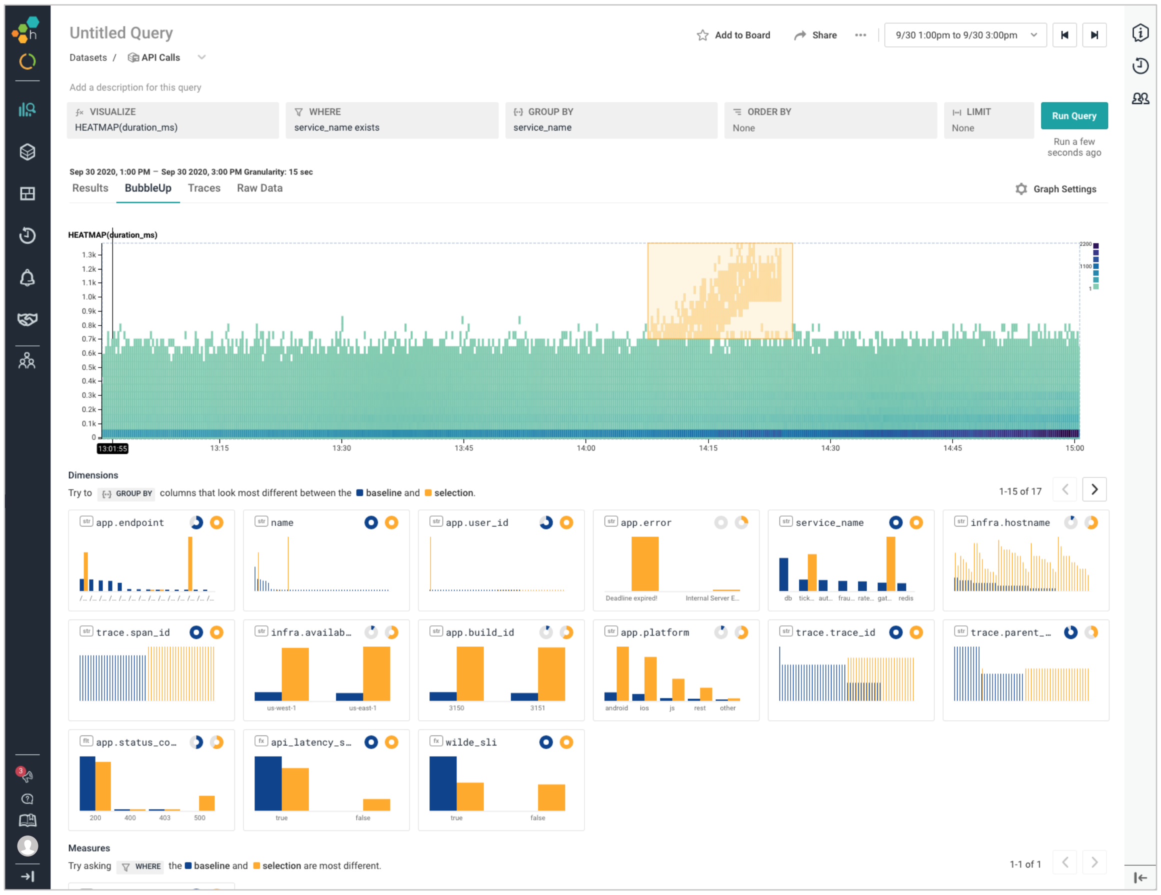
The ability to quickly scan your data to identify which traces will help you troubleshoot production faster is critical. Very few tools allow you to scan billions of data points in seconds. Rapid query response time is only possible with a sophisticated query engine and an underlying database optimized for the ability to perform group-by, aggregates, and filtering. Having an intuitive, flexible query interface allows you to ask new questions again and again across all of your production data. This way you can spend more time writing code and less time troubleshooting or fighting fires.

Introspecting result-sets and interactive charts is really important when debugging and trying to determine where in the code the problem lies. Also with the ability to pull up the set of traces associated with any query you make gives you the power to resolve faster which can often be done without going back to the development team to understand what may be happening for that particular part of the service.

Does the system speed up operator troubleshooting by identifying the correct areas to examine for anomalies?

While many systems tout “AIOps” as a way to identify and surface what’s wrong in production without the need for human insight, there’s nothing quite like the crisp pairing of good old fashioned human intellect alongside machine systems that reduce the search space. Unfortunately, most “AIOps” tools are too noisy and full of false signals. Honeycomb has created BubbleUp, a unique feature to help operators systematically identify what’s causing anomalies.

Traditionally, a system reliability engineer might have to search through logs by hand as they gradually stitch together hypotheses about why production is misbehaving. Using Honeycomb, an SRE team-member can simply point and click on charts that show strange behavior, like latency, and Honeycomb’s BubbleUp feature will highlight charts detailing the likely cause. Ask your observability vendor what they can do for you in terms of identifying the cause of anomalies in your system. With Honeycomb’s incredibly rich support for metadata (up to 64KB per field), BubbleUp can reduce troubleshooting sessions that previously took hours or even days, to just a few minutes.


Untitled Query

Datasets / API Calls

Add a description for this query

VISUALIZE HEATMAP(duration_ms)

WHERE service_name exists

GROUP BY service_name

ORDER BY None

LIMIT None

Run a few seconds ago

Sep 30 2020, 1:00 PM – Sep 30 2020, 3:00 PM Granularity: 15 sec

Dimensions

Try to GROUP BY columns that look most different between the baseline and selection.

Measures

Try asking WHERE the baseline and selection are most different.

Figure 3: BubbleUp’s heat map query result shows events that are different from the baseline (those highlighted blue dots are events that are behaving differently) and right below the heatmap, the bar charts show the likely dimensions that are causing the issue from the selected events in that heatmap (e.g. understand why latency, response time).

03
High-resolution data out of the box

Can the system handle high-resolution metadata alongside the telemetry data?

One of the most useful attributes an observability system can have is the ability to augment the telemetry it emits with high-resolution data. For instance, a very long SQL query may be backing the system up, and the trace span that represents that query should include a normalized version of that lengthy query as an attribute so that an operator can understand why that query is slow and speed it up.

Without this, engineers are stuck with the frustration of knowing something is wrong with dashboards, but endlessly searching and scrolling through logs trying to figure out what that something is. In Honeycomb, the ability to group, aggregate, and filter across all of your data makes troubleshooting rapid. If metrics systems show a "heartbeat" for operations, then Honeycomb's observability is like a high-resolution MRI system that can generate sophisticated images to pinpoint exactly where the source of the problem lies.

Engineers must have a rich visual to understand complex systems and it's no longer adequate to look at logs, metrics and traces in three different tools. Honeycomb believes you need to visualize all three data views in all its multicolored glory. This blog titled "*They aren't pillars, they're lenses*" explains the difference between a single lens view that lacks the depth you need when observing, debugging, and learning.

04
Sophisticated engineering team ergonomics

Does the system support good engineer ergonomics, especially for teams? (e.g, visual history, search, boards, permalinks, Slack integration)

You should seek an observability tool that provides good ergonomics for you and your entire team. This will speed up your ability to troubleshoot and understand production. Honeycomb is designed to resolve production issues with a culture of software ownership in mind. With team on-call rotation, we designed Honeycomb to bring the most junior engineer up to the level of the most seasoned team-member. If it's 3 am and you are dealing with a sev1 issue, you need to start with some hypotheses and follow the analysis loop. It's so much easier when you have a starting point from a fellow team-member.

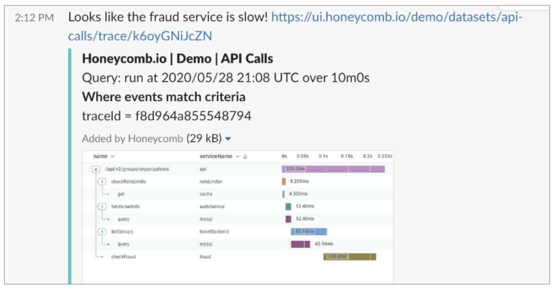
In contrast to a logging system, the core data engine in Honeycomb can sweep through billions of events in seconds, allowing for fluid and iterative analysis because followup queries can be executed so quickly. Instead of going off to get a coffee while your query runs, you can knock through many hypotheses in a rapid debugging session.

Likewise, you can look for rich historical analysis from other team-members, which is a core part of the Honeycomb product. We'd been frustrated by the lack of support for persisting analysis results once the data aged out in other tools, so we built a history feature that makes snapping back to any previous queries a breeze. Your team can use this to learn by example from power users. Likewise, we fit into a modern collaboration suite by supporting things like Slack unfurl. At a rapid glance in your chat system, your team can see previews of query results and underlying traces. As part of the daily workflow, it makes for a smooth, efficient experience keeping teams happy and engaged.

Does the system support OpenTelemetry as a standard way to get data in?

As the industry converges on OpenTelemetry as a common standard uniting the previous standards of OpenTracing and OpenCensus, it's important to ask if your vendor is committed to OpenTelemetry for the long haul. As OpenTelemetry gains popularity, an increasing number of systems (such as infrastructure components underlying your apps) will emit data in this format. Likewise, it will become increasingly likely that engineers are familiar with this standard and want to work with it. Orienting your team towards using OpenTelemetry will boost productivity.

At Honeycomb, we support OpenTelemetry through several exporters and we continue to fit the product to this standard. We think that a common interface uniting metrics and tracing will provide users with more power to choose the right vendor for them, have a shared foundation for tasks such as sampling, and a common language among engineers to export data about what's happening in production. OpenTelemetry is a step in the right direction and you should ask every vendor you evaluate how they currently support this standard, today and in the future.


2:12 PM Looks like the fraud service is slow! https://ui.honeycomb.io/demo/datasets/api-calls/trace/k6oyGNiJcZN

Honeycomb.io | Demo | API Calls
Query: run at 2020/05/28 21:08 UTC over 10m0s
Where events match criteria
traceId = f8d964a855548794
Added by Honeycomb (29 kB)

name	serviceName	0s 0.05s 0.1s 0.15s 0.2s 0.233s
4 /api/v2/groups/organizations	api	233.0ms
1 checkRateLimits	rateLimiter	8.200ms
get	cache	4.300ms
1 fetchUserInfo	authService	12.40ms
query	mysql	12.40ms
1 listGroups	ticketBackend	80.60ms
query	mysql	42.94ms
checkFraud	fraud	120.0ms

Figure 4: Heads-up service is slow and you can see the trace view inside Slack. Jump in and start to investigate.

05
Rapid SLO Iteration

Does the system allow for rapid iteration on SLOs?

Service Level Objectives (SLOs) help you be a better engineer and achieve production excellence with more reliable production systems. They let you define a set of criteria to evaluate your service's overall behavior and track how well you are meeting your specific goals over time. Many services offer an SLO feature, but most also require you to wait for the duration of the SLO (7 days, 2 weeks, 30 days, etc.) for actual data-centric results to trickle in when you make a change or want to publish a new SLO. That's a long time to wait! You can do better with Honeycomb.

The same speedy data engine that powers Honeycomb's rapid query engine allows you to see results instantly when you update an SLO's definition or create a new one, such as decreasing the acceptable latency threshold. Instead of painfully waiting for results to trickle in over time, you can draft several different ideas for SLOs in rapid succession and evaluate the results in real-time to truly understand how the service is behaving and performing right now. This creates a positive feedback loop, encouraging you and your team to create more SLOs and iterate as you go.

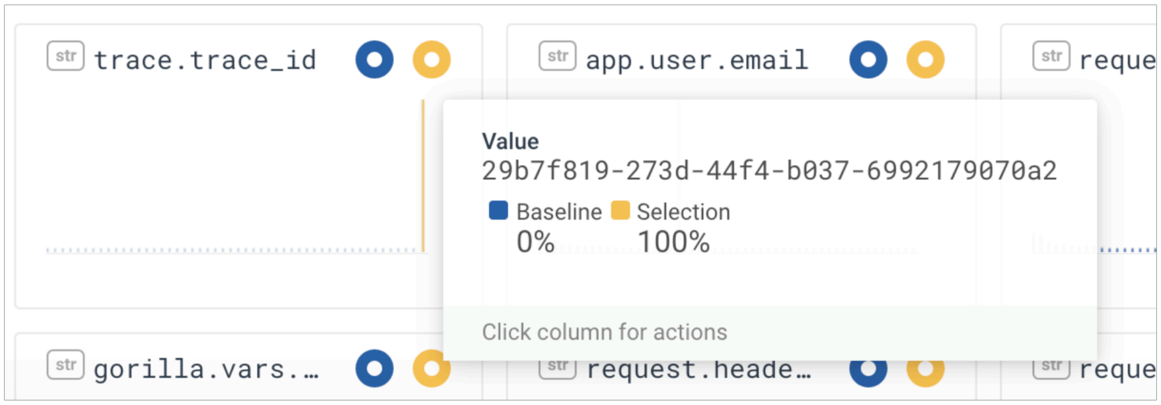
Ultimately the goal of establishing SLOs is to understand how your users are interacting with the service and how it's behaving based on what is acceptable and expected. Think of SLOs as a way to tell you how the service is doing for end-users. Honeycomb's budget burn-down charts tell you exactly how the service is at that moment in time and how much budget you have for the remaining time-frame. In this way, you prioritize team workload and don't waste time troubleshooting or trying to resolve an issue that can wait until the next day or later that week.

06

Actionable insights for error budget burn-down

Does the system automatically identify what's wrong with requests that burn the error budget for SLOs?

Honeycomb SLOs help you track how well your systems are performing over time, and ensure that you are continually improving the reliability and stability of the service. Likewise, Burn Alerts will alert you and your team when you are in danger of burning through the rest of your budget. Ultimately, this will allow you to ship more changes with confidence as you will be able to proactively curtail problems from recent changes when needed.



We don't stop there. Any events that are violating the SLIs you have configured to power your agreed SLOs are automatically integrated into Honeycomb's BubbleUp feature that explains anomalies. This helps to rapidly determine the cause of what's eating into your budget for availability, latency, and reliability, assuming those are important SLOs for your users.

These insights from BubbleUp even include which specific traces are using up your budget. Jumping right to the specific trace(s) that may be causing your systems to be unreliable and your goals to be out of reach is blazingly fast and incredibly reliable.

Optimize for Better Outcomes

These specific features and capabilities are absolutely key when observing complex services at scale. Engineering teams who code, ship new releases, and maintain site reliability and performance need better tools to understand what is happening right now. Teams need to spend less time guessing and wasting energy troubleshooting when they can access the power of observability. Engineers need to focus on the most important and valuable part of their craft—designing and creating—which means less time debugging, fixing, and refactoring.

With a culture of shared ownership, a tool that provides hi-res visibility to all streamlines effort and yields happy devs and happy customers. Predictably ship new code and avoid a cycle of fear. We hope this buying guide was helpful and we'd love to hear any feedback you wish to share.

Further Resources to Learn about Observability

Distributed Tracing

Distributed Tracing, A Guide for Microservices and More

By Nathan LeClaire, Sales Engineer at Honeycomb

What is Observability and Why Should I Care?

A Next Step Beyond Test Driven Development (TDD) Blog

By Charity Majors, CTO & CoFounder at Honeycomb

Culture of Observability & Shared Ownership

Developing a Culture of Observability Guide

By the Honeycomb team

OpenTelemetry & How Honeycomb Supports this Open Standard

OpenTelemetry: New Honeycomb Exporters Blog

By Paul Osman, Lead Instrumentation Engineer at Honeycomb

Importance of Instrumentation

Incremental Instrumentation Beyond the Beeline Blog

By Shelby Spees, Developer Advocate at Honeycomb

SLOs and Where to Start

Get Started with SLOs. Build One Simple SLO Webinar

By Nathan LeClaire, Sales Engineer at Honeycomb and
Josh Hull, Site Reliability Engineering Lead at Clover Health

Ready to dive in? Start with the Free Tier

About the author

Nathan LeClaire (@dotpem) is a Go programmer and author who enjoys open source and DevOps. He lives in San Francisco, CA and currently works as a sales engineer for Honeycomb.

About Honeycomb

Honeycomb provides observability 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 and LinkedIn. Visit us at Honeycomb.io