
OpenTelemetry Myth Busters
Investigating Common Misconceptions
splunk>
a CISCO company

The science of observability

Remember learning about the scientific method? In case you haven't thought about it in depth since your middle school science fair, it's a series of steps for testing a scientific theory. It starts with asking a question. From there, you conduct research, develop a hypothesis, experiment, analyze the data, and draw conclusions.

On the popular TV show *MythBusters*, researchers demonstrated the scientific method in every episode as they worked to validate or disprove common myths or urban legends. They proved that water heaters can explode if the pressure inside is high enough, but the explosion isn't as destructive as the myth suggests. They attached rockets to a car, disproving the myth that it would take off. While you're probably not experimenting with water heaters or strapping rockets to a car as they did back in the day, you're likely following the steps of the scientific method to answer questions within your observability practice.

For example, if you notice that your users are experiencing higher-than-normal latency, you do some research to understand why. You may hypothesize that slow or unstable network connections are causing the problem. Then, you experiment to test this hypothesis, gather data, identify a solution, and bring it back to your team.

Data is a key part of any science experiment — and honing your organization's observability practice. In observability, data in the form of metrics, logs, and traces is necessary for testing hypotheses and identifying problems. However, gathering and interpreting data can be challenging when all of this data is collected with disparate tools, leading to less accurate and actionable insights.

Many organizations are turning to OpenTelemetry (OTel) to solve this challenge. OTel provides the most flexible solution for transferring observability data from an application into an observability system. But like any other emerging trend or scientific theory, there are still some myths and misunderstandings surrounding it.

"OTel is too difficult to implement," one engineer might say. "It's strictly for cloud-native environments," suggests another.

So, are these OTel myths true? In the spirit of science and TV show nostalgia, let's start our own experiment to find out whether we can confirm these myths — or bust them.

MYTH #1

OTel is overly complex and difficult to implement

Kai, a software engineer, and Jaz, an SRE, have worked together as engineers for years. Their team struggles with poor visibility in their tech environment, making it difficult to identify the root cause of issues. Because of their slow MTTx, their team regularly experiences extended on-call hours. To help with this, Jaz suggests adopting OpenTelemetry.

“Okay, Jaz, I get it. You’re all about OpenTelemetry, but integrating it with our applications and training the team sounds like a hassle,” Kai says. “Do you think it’s really worth it?”

“It’s just like any other infrastructure,” Jaz says. “Once it’s up and running, we’ll be smooth sailing. It’s only going to get easier as time goes on. And let’s be honest, what’s tougher than dealing with on-call nightmares at 3 a.m.?”

Kai sighs. “You’ve got a point. These late-night calls have been awful. I’m open to seeing if you’re onto something, but I’m still concerned that setting it up could be a pain.”

“I get it. But powerful things are complex by nature,” Jaz says. “OpenTelemetry would let us meet our data residency, privacy, and data tiering requirements through processing pipelines. We can manipulate data however we want and change that easily later after the instrumentation work is done.”

Kai and Jaz agree to work together to test out just how complex and difficult implementing OpenTelemetry really is. With automatic instrumentation, it didn’t take long to set up — but once complete, they immediately notice improved visibility, faster troubleshooting, and a clearer view of overall system health.

“That wasn’t as bad as I thought,” Kai says. “Instrumenting our apps with OTel turned out to be pretty manageable, and it was well worth the effort. Now that we’re able to address performance issues more proactively, I haven’t gotten a single late-night call. It’s a win.”

MYTH #2

OTel is only for cloud-native environments

Rae, a systems administrator, is talking to Max, a DevOps engineer, in the office about their team's issues with incomplete and inconsistent data. Max suggests that OTel is a unified, lightweight collector that could help the team collect metrics, logs, traces, and more, but Rae is skeptical.

"OpenTelemetry sounds promising, but I don't think it would even work for our three-tier environment," Rae says. "We're on-prem, and since OpenTelemetry is a Cloud Native Computing Foundation project, it must only be for cloud-native systems. Plus, so much of their advertised support is for cloud-native tech."

"Actually, OTel works for both on-prem and cloud environments," Max says. "It's definitely useful for distributed systems and microservices architectures, but it can also be valuable for applications like ours. Regardless of how complex a system is, I've seen it help with understanding performance and troubleshooting."

A few weeks later, Rae decides to follow Max's recommendation and instrument the backend of their three-tier application with OpenTelemetry to confirm or bust the myth.

"I was definitely skeptical, but I'm surprised to say Max was right," Rae says. "We started using OpenTelemetry for the backend, and it ended up working well for the rest of our three-tier application too. Now, our team has more streamlined and unified observability with consistent data across metrics, logs, and traces, and we can integrate observability of our three-tier app with observability for our cloud-native mobile application."

MYTH #3

The product is immature, and troubleshooting is difficult

Drew, a software engineer, is pitching OpenTelemetry as a replacement for their team's legacy proprietary APM system. Nia, the team manager, is pushing back.

"OpenTelemetry isn't ready for prime time yet," Nia says. "It's not even 1.0 — everything is in alpha and beta. Everyone's talking about it these days, but I used it at the startup I used to work for and could never figure out what was going on."

"There is great documentation, conversation, and community around OTel today," Drew says. "Besides, versions don't hold as much weight anymore. Products you use every day have been in beta since they launched. Plus, it's the best solution out there. Many observability teams and practically every vendor have jumped on the OTel train."

"I don't know, Drew. I've also heard OTel is only for application tracing," Nia says.

"I've heard folks say that as well, but they may be overlooking its capabilities for logs and metrics," Drew says. "OTel provides standards and APIs for collecting and formatting various types of telemetry data from all sorts of programming languages and frameworks. New types are being added too — profiling is coming soon."

With Drew's encouragement, Nia learns more about OpenTelemetry's support for profiling, which gives insights into resource (i.e. CPU and memory) utilization at a code level. Since so many of their peers at other companies are getting on board, they decide it's time to give it a go.

Fast forward to a few months later: "It's been a few months since we've instrumented our apps with OTel, and looking back, I understand why Nia thought that way," Drew says. "The project is still relatively young, but there's a huge community around it, like the CNCF Slack. Just because some components are still in beta hasn't meant they're unreliable, either."

"I agree," Nia says. "The fact that everybody else is on board has made Stack Overflow really busy with OTel info too. It's had some growing pains, but so many people use OTel now that it's easy to find what we need. Plus, it has reduced our tooling complexity and has made it easier to scale observability."

Myth BUSTED

MYTH #4

OTel is expensive

Charlie, application engineering director, and Rory, software engineering manager, have decided that their engineering team needs a standardized approach to monitoring and troubleshooting. Rory suggests OpenTelemetry.

“I know you think we should be using OpenTelemetry, Rory. But the thing is, OTel is so expensive,” Charlie says. “I saw some ads about how pricey it is.”

“Some vendors are trying to push that misconception,” Rory says. “Of course, there are time costs associated with it, just like with anything else. Instrumentation work is toil, but you have to do it, so it’s better just to do it once and do it right. If we instrument with OpenTelemetry, we won’t ever have to do it again even if we later switch observability platforms. We can just make some changes to our configuration to send data to a new platform instead of having to rip-and-replace agents like we would with proprietary solutions.”

Charlie later decides it’s worth the risk to run an OpenTelemetry experiment by instrumenting just one microservice to see if the high costs are just a myth after all. After the instrumentation process, Charlie realizes that Rory was correct about the time costs, but now they know they’ll always have observability for their team’s applications no matter what vendor is picked for their observability backend. That knowledge, and the flexibility to process and manipulate observability data effortlessly, were worth the toil.

“OpenTelemetry itself didn’t end up costing anything. There are no licensing costs,” Charlie says. “In the long run, the benefits are greater than the time investment required.”

MYTH #5

If our legacy system is working, there's no need to change

"I don't get why we would spend a bunch of time and money on this observability fad when we have ping checks," Sam says.

"There's so much more than just uptime that matters," Sage says. "Our apps are getting more complex, and we need visibility. Ping checks don't give us the insights we need. With OpenTelemetry, we'd have unified observability with detailed tracing and metrics, which would help us diagnose and resolve issues more quickly."

"But isn't our current system good enough for what we need now? I don't know if the learning curve with OTel is worth the challenge." Sam asks.

"Maybe so," Sage says. "But is just 'good enough' what our customers are looking for in our product long-term? They expect more from us than that. Our product doesn't exist in a vacuum. Its quality depends on the processes supporting it to some degree, including our ability to observe the product's performance. After all, slow is the new down."

"True," Sam says. "We can give it a try. I don't want a subpar observability solution to result in a subpar product. Our customers do hold us to a higher standard."

Since implementing OpenTelemetry, Sage and Sam's team has better visibility into far more than just uptime — they can now see latency, error rates, resource utilization, anomaly detection, and more. This comprehensive view has helped them speed up MTTR and boost user satisfaction.

MYTH #6

It’s difficult to manage

Arlo, a platform engineer, and Billie, an SRE, are discussing OpenTelemetry as an option for their team, but Arlo is experiencing a learning curve around the complex configurations.

“OTel looks too difficult to manage,” Arlo says. “When I look at the configs, it looks like there is no way to orchestrate this. We don’t have centralized control over the collector configurations. It looks like there’s a lot of components I’ll have to manually keep track of on all our different types of infrastructure. I get that it’s ideal to have observability built-in as part of our platform, but I need to be able to centrally see what’s running alongside all our different apps.”

“You’re right, it still is challenging. But the team at Splunk is working on that going forward,” Billie says. “There are features in the works to inspect the config of collectors across a deployment and to update and manage those centrally. Even right now, though, all the configs are text files, so they can be managed like you deploy your other configs. Centralized config management can be done using your current deployment or orchestration system, like Salt, Chef, or Puppet.”

Myth: Busted?

This one has some truth to it, but like the rest of the project, improvements are coming rapidly.

OpenTelemetry: The future of observability

While some myths are still swirling around OpenTelemetry, we hope these stories helped you gain a better understanding. Like any other solution, OTel has its challenges, but it's essential for your observability practice. As more and more organizations adopt it, they discover that the benefits far outweigh the limitations as they experience faster MTTR, unified observability without vendor lock-in, limitless flexibility and control over their application data, enhanced scalability and troubleshooting, and more. Plus, with a growing community of developers and contributors, you can still jump on board and become an expert.

OpenTelemetry provides the instrumentation and standardization you need to gather telemetry data, but you also need a system like Splunk Observability Cloud to analyze and visualize the data. Splunk is a top contributor to the OpenTelemetry project, and OpenTelemetry is native to Splunk Observability Cloud. This means Splunk customers benefit from quick delivery of modern OTel framework and language support, speedier time to value, and lack of vendor lock-in.

Learn more in How OpenTelemetry Builds a Robust Observability Practice.

splunk> a CISCO company

24_CMP_ebook_otel-myth-busters-investigating-common-misconceptions_v9