StackState

The Ultimate Guide to Telemetry

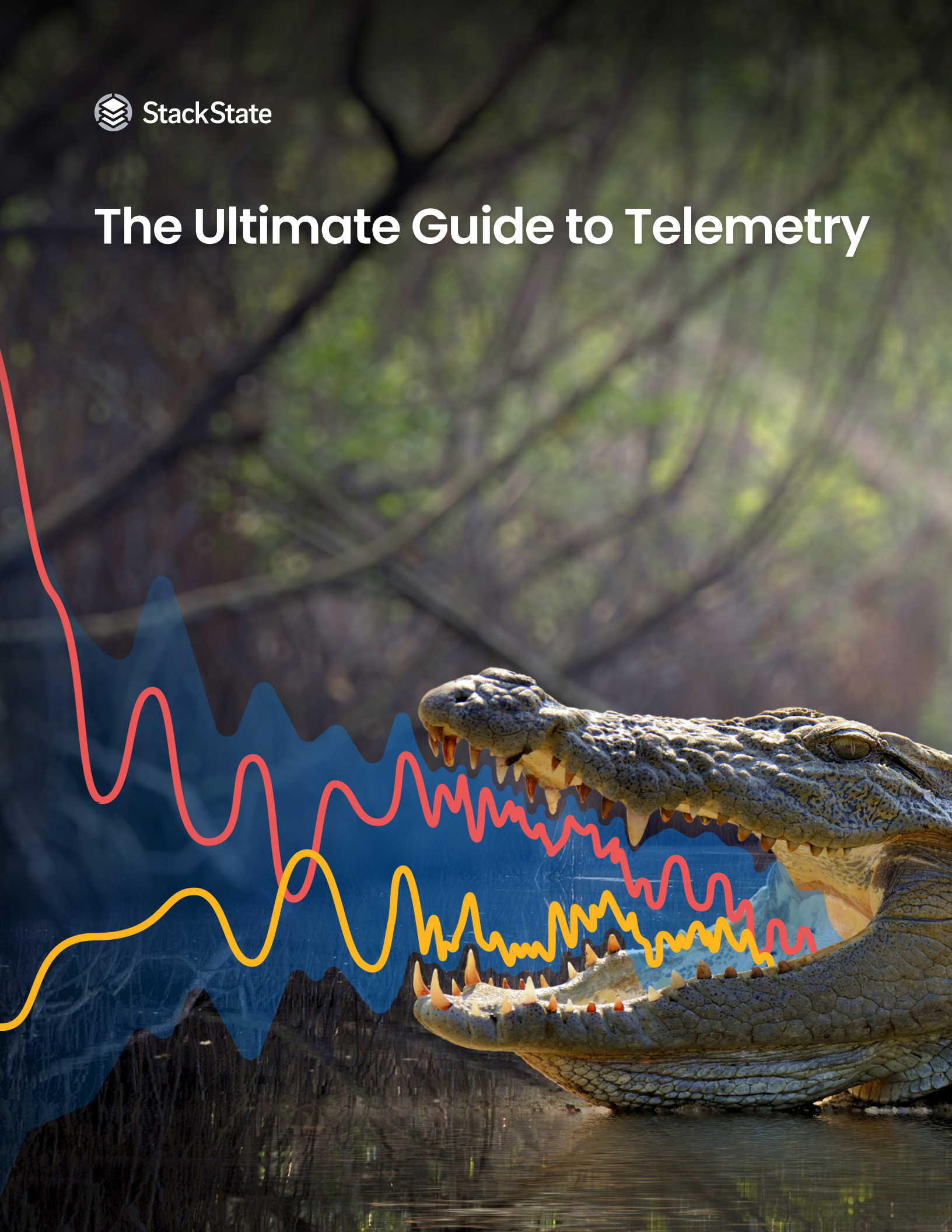
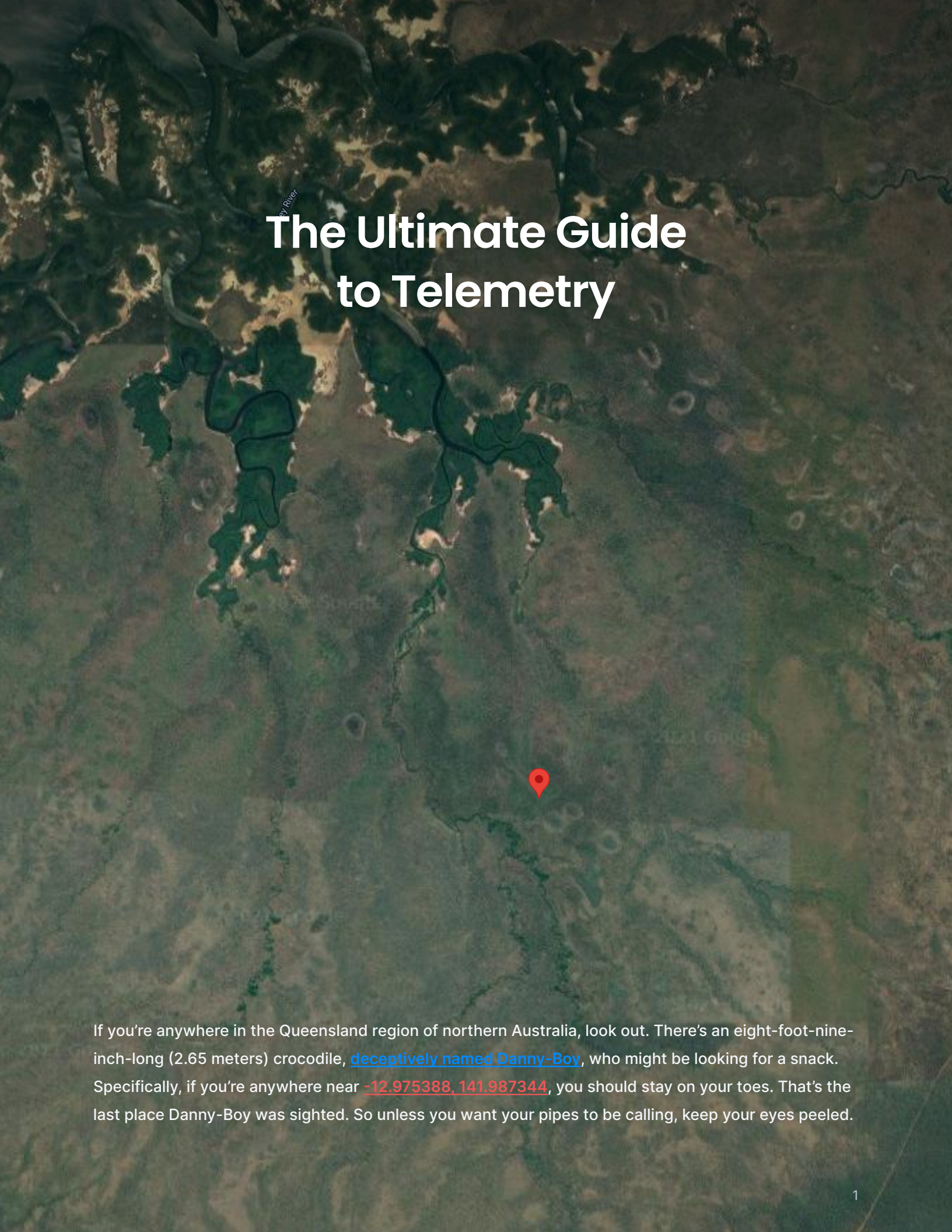


Table of Contents



The Ultimate Guide to Telemetry

If you're anywhere in the Queensland region of northern Australia, look out. There's an eight-foot-nine-inch-long (2.65 meters) crocodile, deceptively named Danny-Boy, who might be looking for a snack. Specifically, if you're anywhere near -12.975388, 141.987344, you should stay on your toes. That's the last place Danny-Boy was sighted. So unless you want your pipes to be calling, keep your eyes peeled.

Thanks to telemetry, anyone with an internet connection can get location information about Mr. Danny-Boy, a crocodile tagged by the Australia Zoo back in 2019. Researchers can use telemetry data to study where Danny-Boy goes, how long he spends there, how much time he spends around other crocs and even his feeding habits.

A small device was inserted in Danny-Boy's back and it sends data back to the Australia Zoo, where it is used to deduce important information about Danny-Boy's habits.

Although the device perched atop Danny-Boy is a very simple example, that's exactly how telemetry works. In IT systems, however, you can get a long list of critical data sent to you. And, just like Danny-Boy's trackers, you can use this data to avoid getting "bitten" by system failures.

What Is Telemetry?

Telemetry involves a system of collecting data about devices and network components and then transmitting it to a central location for analysis and application. While it's been in use since the days of the steam engine and has applications extending far beyond tracking cute, cuddly crocs, telemetry has recently assumed a major role in the observability and optimization of IT systems.

While telemetry can play a vital role in an in-house IT infrastructure, its application applies equally well to cloud-based systems, as well as an innumerable assortment of Internet of Things (IoT) devices. IoT devices can even use telemetric data to self-correct and optimize either their own performance or that of those wearing, watching, reading, listening to or working with them.

For example, a Fitbit can keep track of the amount of calories you burn while running, hiking, biking, doing circuit training or on the elliptical. It can also track your pulse, blood pressure, and other biometric data. This information in and of itself, has undeniable value. However, the application of the data points that come from a Fitbit is far more exciting.

For instance, someone can use a Fitbit to **ascertain whether or not they have sleep apnea.** You can also determine the severity of asthma using data the Fitbit provides regarding the oxygenation levels of your bloodstream.

Telemetry can be applied in similar ways for enterprise or small to medium-sized business operations. The data the telemetric system collects has value, but the truly exciting insights stem from how you can use the data to determine the health and performance of your system, as well as how to optimize performance.

Sources and Input

Telemetry begins with deciding which sources you want to gather information about and then the kinds of inputs that will best fit your objectives.

Sources

You can think of your IT infrastructure as a human body. Your various processes, components and systems are like the arms, legs, hands, feet, fingertips and toes of the "body" of your network. Your observability platform works like a nervous system, connecting sensors in the extremities of your system's body, gathering data from each one and presenting for use in troubleshooting, root cause analysis and system optimization.

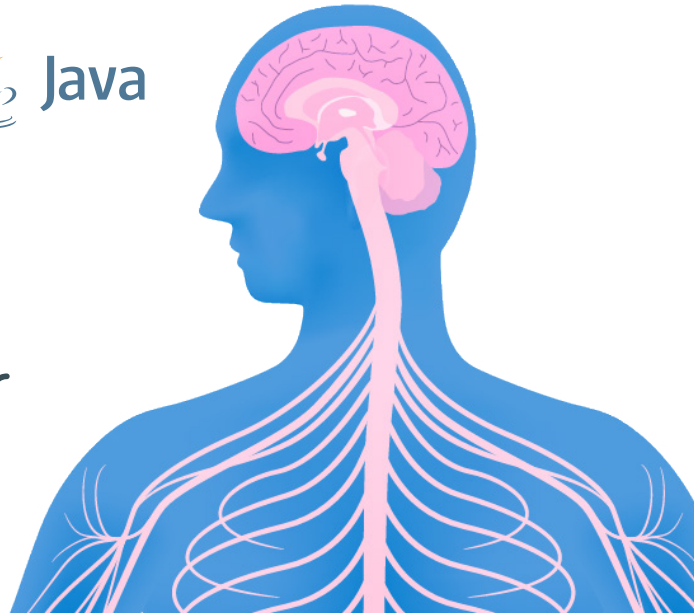
If a component experiences an issue, your telemetry sends a message to the observability platform, which works like a brain to help you ascertain the problem. With the proper use of telemetry, you can differentiate problems from each other and instantly figure out root causes. If you stub your toe in the dark, you can tell whether you kicked the couch or your three-year-old's toy truck. Similarly, telemetry can enable you to instantly differentiate the causes of different problems. This process begins, however, with choosing the right sources.

In a typical, simple setup, your sources may include:

- Docker containers
- Servers
- Java processes
- Databases

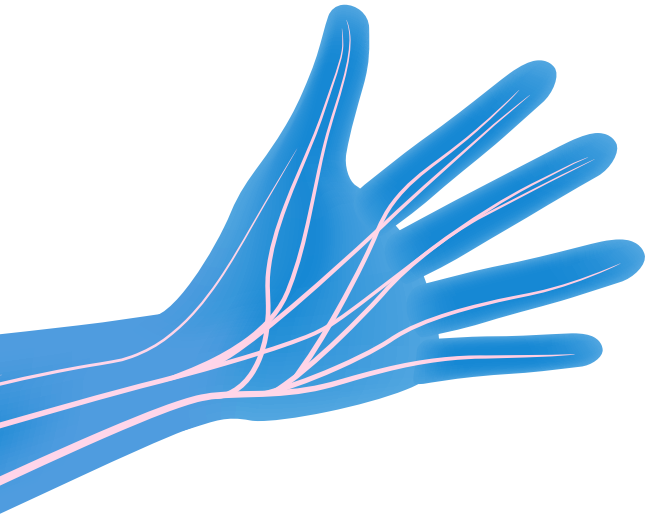
These elements all interact with each other. As a result, you can reap a few different benefits:

- You can use telemetric data to see how one system is affecting another.
- You can intentionally break a state of one source and use telemetry to see how a broken state impacts the overall process.
- You can trace the source of an issue by looking at the telemetric data produced by each broken process. The source at the root of the problem would have broken before those subsequently affected.



Inputs

Inputs refer to the types of data you collect from a source. Another way of looking at it is the “place” from where data streams. If you think of your hand as a source, you can divide things your hand can feel into inputs. If you put your hand in a pot of liquid, your nerves beneath your skin can send several different data points to the observability system of your brain: temperature of the liquid, whether it’s thick or thin, smooth or clumpy, watery or like Cream of Wheat.



It’s similar to the sources within your telemetric system. Each source can produce different kinds of inputs, and these can be used to diagnose issues and make individual components or the entire system function better. Because the volume and actionability of your telemetry system depend on the relevance of the data you collect, you need to choose the right inputs. For instance, you may want to gather data regarding how much heap, or memory space, a Java process is using. This will give you data that can be leveraged in a variety of ways. By analyzing heap, you can take steps to determine:

- Which processes are consuming the most memory at any given time
- Ways to prevent critical resources from being overburdened when they are needed by other processes
- How to appropriately size the heap without impacting performance

How Input Choice Can Improve Processes

Choosing your inputs can go a long way to preventing mistakes based on false assumptions. Returning to the Java example, suppose you're concerned about how to size your memory heap. You're going back and forth about whether to go for a larger heap or a smaller one. Your primary worry is that if the garbage collection process results in inordinately large pauses, the end user's experience will be negatively impacted. (Garbage collection refers to when the system does an audit of memory that's not being used and then reallocates it to processes that need it.)

Big heaps mean less frequent garbage collection and longer pauses while smaller heaps mean more frequent garbage collection and shorter pauses. If the priority is shorter pauses, it may seem logical to go for a smaller heap.

But how do you know that's the case with your specific setup? Without telemetry, you don't. True, you can test it out using trial and error, but that may consume valuable time. By taking telemetric data about the amount of heap being used, you can run your processes using different usage scenarios, record the data sent back through your telemetric system and analyze the results. In this way, you can make an informed decision.

How Input Choice Helps Enhance Root Cause Analysis

The applications of strategic input choice extend to root cause analysis equally well. Circling back to the Java example, using heap as an input not only benefits decisions regarding heap size, but it can also be used to ascertain the sources of problems.

If, higher up in your observability structure, a performance delay is observed, the data from your heap telemetry point can provide helpful insights. Suppose you have a Java virtual machine (JVM) that's running a business-critical application. The application is using an object that you know has the potential to consume a lot of heap or memory space. Is the performance delay due to the memory being consumed by the object? Using your telemetry data, you can get a definitive answer.

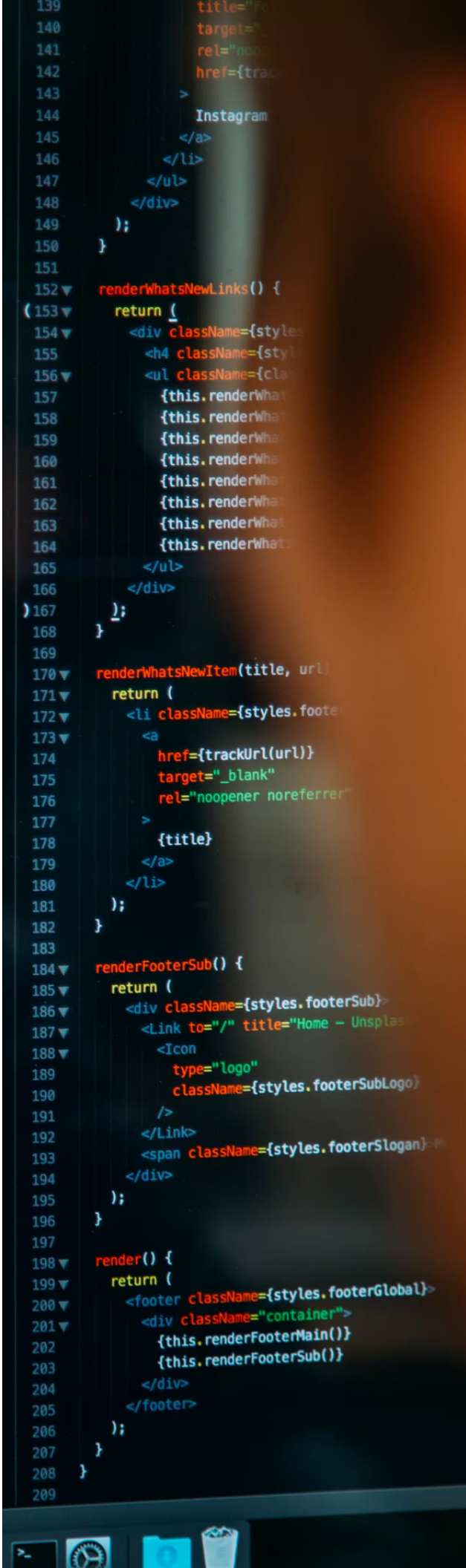
This way, input choice can pay considerable dividends when it comes to figuring out the root causes of issues or failures.

Output

IT observability can be greatly enhanced by deriving specific sets of insights using telemetry output data. In a topology-powered observability (TPO) model, analysis of strategically chosen outputs can make it far easier to relate changes to incidents. In addition, the performance and state of your different apps, components and processes can be evaluated on a consistent basis using telemetric data.

In this way, the kinds of insights you're able to glean are virtually limitless. Similar to a Fitbit, while the data sets the Fitbit is capable of producing are impressive, the different ways they can be used to pinpoint health issues are what truly add value.

For your IT stack, choosing the right outputs is similarly a crucial step in realizing the value of your telemetry system. Here are some basic examples of telemetry outputs and how they can be used to generate helpful insights.



Requests Sent to a Web App

A telemetry system can log data related to the kinds of requests that get sent to your web app. A telemetry system can also collect information regarding when the request was made and from where it originated. Request information kept in a log can provide crucial data which can be used to analyze things such as user behavior and the effectiveness of your web app. You can use this information to analyze things such as user behavior and the effectiveness of your web app.

User Behavior

Telemetry that analyzes the requests sent to a web app can go a long way towards figuring out how and when users are engaging with your app. For example, you can track:

- How often are users sending requests to your app?
- Are their requests happening infrequently? If so, what could be causing the lack of requests?
- Are there certain times of the day when more requests tend to come in? If so, would it be possible—or advisable—to adjust when and how other resources are used during those times of the day?
- Have requests suddenly stopped? This could signify a significant issue elsewhere.
- How were requests affected during certain events that were localized to specific areas? For instance, if there was a natural disaster in an area, did the request volume fluctuate significantly? Or if there was an outage that impacted a key data center, how did that affect the volume and timing of the requests?

This and other data can be used to observe the performance of your web app and figure out ways to improve it. You can also use this kind of telemetric data to explore ways to adjust your marketing strategies based on request volume. Further, you can adjust the ways in which you educate business users if you notice a lack of app usage.

The Effectiveness of Your Web App

Whether or not your web app is effective can be judged using telemetric data showing which aspects of the app are being used the most. For example, you can observe:

- **The settings users prefer to select:** If there are several settings that users are choosing not to use over significant periods of time, you can use this data to reconsider their effectiveness.
- **Display types:** If some display types are being chosen by users more than others, you can use that telemetry data to replicate the success of those displays. You could also draw the conclusion that other displays may be less effective, and you could, in turn, limit the selection of display options.
- **Methods of input:** If users have a choice of input modalities, you can use telemetry to examine which ones are the most popular. In this way, those that are used less can be marked for improvement or replacement. Perhaps they're too hard to use and you should divest resources to support those modalities and use them to support more successful methods of entering information.

How Telemetry Outputs Help Diagnose Issues

As is the case with a Fitbit or even Danny-Boy the crocodile, telemetry's primary benefit is found in a combination of the data it gleans and how that data is used to generate insights. In a setup that combines cloud native applications in both AWS and Kubernetes, for example, telemetry can help diagnose issues faster than they could have otherwise been addressed.

With a relationship-based observability platform like StackState, you can gain visibility into both your AWS and Kubernetes platforms in a single pane of glass. The alerts generated in connection with issues and failures are fine-tuned using telemetry.

For instance, suppose an application in Kubernetes has failed. Your telemetry setup is designed to monitor each microservice as well as the processes that use them. If a database that is used by multiple objects suffers a failure due to an internal failure, your telemetry can come to the rescue.

If, for example, four processes fail and they not only use the same database but also have other dependencies in common, telemetry can eliminate the guessing game. If your telemetry is producing outputs regarding the performance of the database, you don't have to sit around trying to figure out where the problem came from. The database issue can be reported directly to the topology-powered observability platform.

What Is OpenTelemetry?

OpenTelemetry is a cloud native, vendor-neutral collection of application programming interfaces (APIs), tools and software development kits (SDKs). It's used to generate telemetry data, as well as collect and export it. You can then use what OpenTelemetry produces to gain insights into the way your software performs or behaves.

OpenTelemetry supports a variety of languages, such as:

OpenTelemetry also integrates with many common libraries and frameworks, including:

With OpenTelemetry, you can forward telemetric data from your software and services to analysis tools. This allows you to gather, organize and analyze telemetry data to gain insights.

Why Is Telemetry Important for IT Observability?

Telemetry is an important component of IT observability because it provides admins with accurate, actionable data. Further, because telemetry can be set up to bring data down to a granular level, you can zoom into important details in your IT environment.

With a well-chosen array of telemetric data points, you can gain deep visibility into specific services. By doing so, you are in a better position to:

- Figure out not just the component that houses the root cause but what happened within its process that resulted in the issue in the first place.
- Observe how failures on a granular level impact the performance of a system as a whole.
- Better understand how the performance data of different services impacts the end-user experience.

Overcoming Telemetry Limitations

Is Big Brother Watching?!

One of the biggest limitations of telemetry is that some of it is dependent on the user agreeing to allow telemetric data to be taken while they use an app. This can present a significant challenge because in situations where the user is allowed to opt out, they may feel safer doing this than allowing someone to monitor their usage.

In a sense, they fear being spied on by Big Brother.

How to Overcome the Big Brother Concern

One way to try to get past users' fear of being spied on is to be completely upfront regarding the telemetric data you're collecting. But make your statement regarding the info you are collecting easy to understand and quick to read.

For example, you can outline specific things the user may do that could result in data getting sent to your observability platform. That way, the user may think, "Oh, well, them knowing that isn't so bad." You could also explain how collecting the data benefits the user. A statement such as, "To maximize your uptime and quickly address any software issues, our system collects usage data." Then, when they're offered the option to accept or reject, they may be more likely to click the agree button.

Too Many False Alerts

In an attempt to design a more comprehensive system, admins may set up telemetry systems that appear to over-report issues. This could result in false alarms, infecting your IT team with a "boy who cried wolf" mentality. When there's a real problem, those who can handle it have heard "Wolf!" so many times without any consequence that they become inured to the cry.

How to Overcome the False Alert Obstacle

There are a few things you can do to limit the number of false alerts. One is to fine-tune your telemetry system so it only reports issues that indicate that a business-interrupting failure is imminent.

You can also adjust the way telemetric data is processed. For instance, you can program in a rule that dictates an alarm only goes off if a certain number of alerts are triggered over the course of a specific time frame. This could work because, in some cases, the more frequent the issue, the more severe the problem.

How Telemetry Is Used with StackState

Telemetry as One of StackState's 4 T's

StackState uses telemetry in conjunction with three other factors to bolster our topology-powered observability model. Telemetry works as one of the 4 T's:

- Topology
- Telemetry
- Tracing
- Time

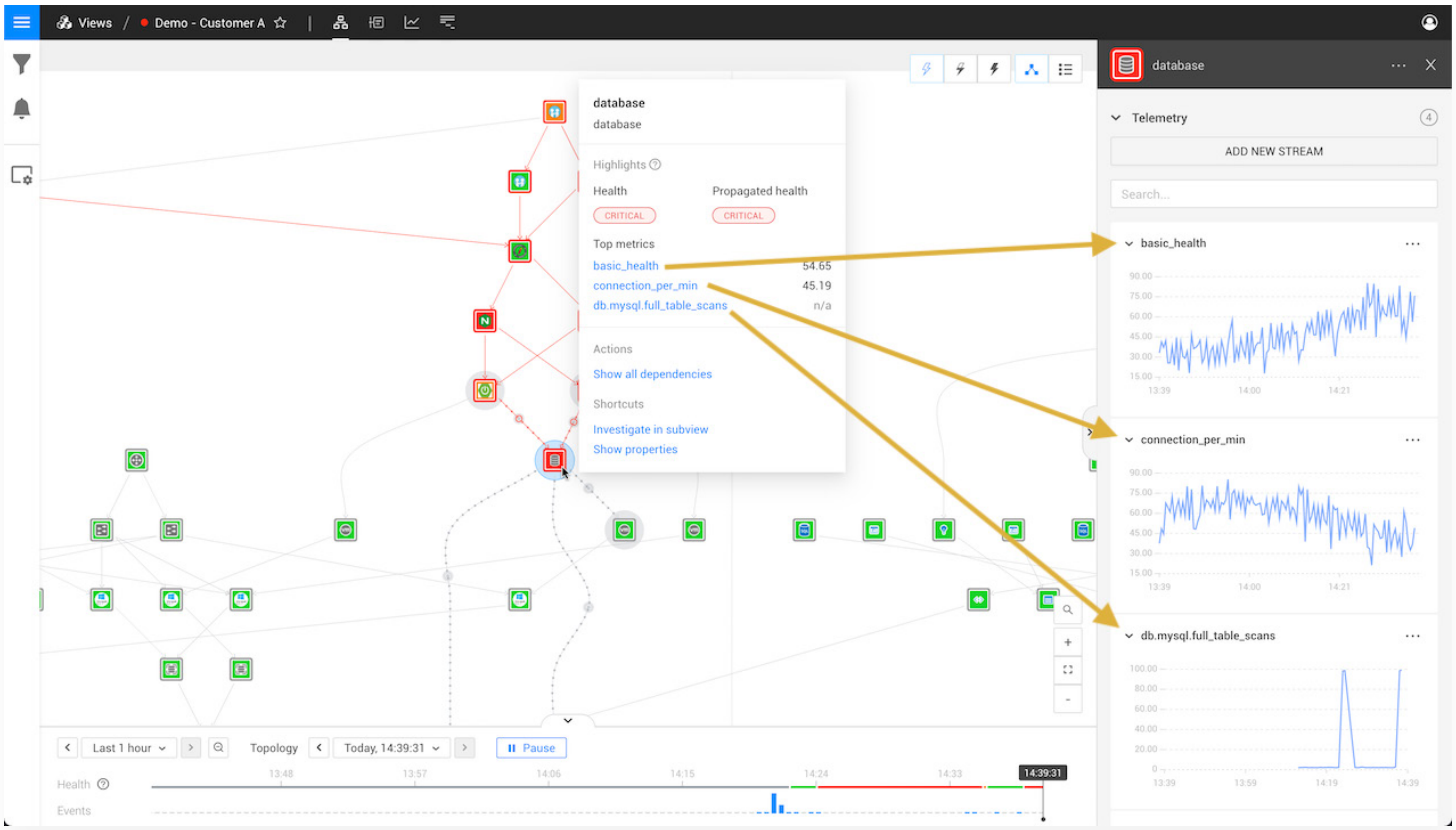
Within this model, **topology** maps out the topology of your system, showing the relationships between different components, presenting them as a map in a graphical user interface (GUI).

Time is another factor mapped within the GUI, represented as a single vertical line running along the Z-axis, alongside the X and Y axes of the topology map.

Tracing gives you end-to-end insights at the code level. Because tracing is integrated with the topology and time features, it can be used to pinpoint how issues impacted elements of the topology at specific moments.

Telemetry is used to send a message to the observability model regarding exactly which component suffered a failure. The observability system then creates a snapshot of the system in its healthy state—the moment before telemetry indicated the failure. This snapshot is automatically associated with the moment in time it occurred.

Thereafter, the telemetry system continues to send data regarding the health of the system and its individual components, allowing admins to see how fixes are impacting system performance or function.

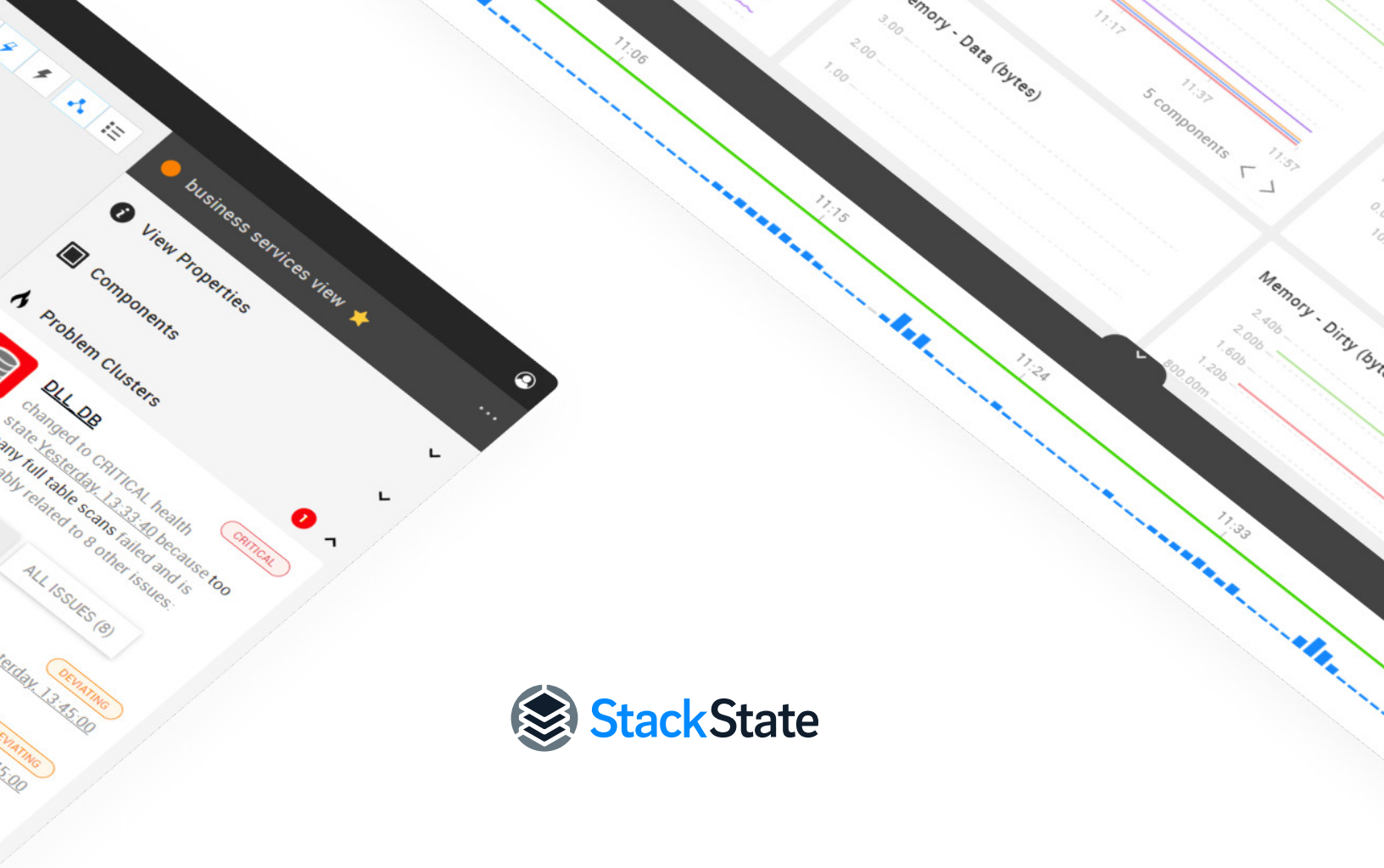


StackState dashboard showing top health metrics including corresponding telemetry streams.

StackState's Observability Platform Maximizes the Potential of Telemetry

Telemetry provides detailed info regarding the performance of specific components of your system. It uses sources and input, allowing you to decide which components, apps and services are observed. It then leverages the data from these inputs to help you derive business-critical insights.

StackState's telemetry tools allow you to monitor the performance of many elements of your system at the same time. The system can then automatically report events, time-stamping them and presenting them in a GUI that's easy to read. You can use these in root cause analysis, to track the impact of changes and to improve the efficiency and function of your system. Telemetry has gone from steam engines to crocodiles to your wristwatch. With StackState, telemetry goes a step further by combining with Topology, Tracing and Time to support your business, its end users and the success of your organization.



StackState

Make the Most of Your Logs, Metrics and Traces

Topology-powered observability correlates your telemetry data over time and combines it into one clear overview. The result? A complete picture of the state of your stack at any moment in time, so that you can prevent issues and crush resolution time.

For more information, play in our sandbox or schedule a personalized demo with one of our solution engineers.