A Beginner's Guide to

Observability

Cutting through the complexity to learn what your systems, services and apps are really doing

splunk>
turn data into doing™

Observability has been called everything from a trendy tech buzzword to a "monitoring-on-steroids" must-have. The truth is more involved — especially given the increased complexity of modern infrastructure and the undisputed need for better monitoring everywhere in the stack. Microservices, containerization, serverless — all accelerate development velocity and provide important business benefits, but also multiply complexity and decrease visibility.

A simple three-tier service in the old days may now be run by as many as 50 microservices deployed to multiple cloud providers. Some "microservices" may actually be calls to serverless functions. This makes observing your entire application far more difficult.

Additionally, teams requiring operational visibility have expanded beyond sysadmins and ITOps analysts. With modern DevOps models, developers are taking greater ownership of knowing what's going on for a better customer experience. To effectively do this, all roles need visibility inside their entire architecture — from infrastructure through applications all the way out to third-party APIs — to fix (and ideally to prevent) problems.

Observability goes beyond mere monitoring (even of very complicated infrastructures) and is instead about building visibility into every layer of your business. Increased visibility gives everyone invested in the business greater insight into issues and user experience, and creates more time for more strategic initiatives, instead of firefighting issues. It's also critical to the overall success of site reliability engineering (SRE) or DevOps organization models. In modern DevOps development processes, developers need visibility into operational performance as much as traditional ops teams or SREs.

In this guide, we'll define what observability is and what it takes to achieve it. We'll also give some examples of observability in action and guidance for what to look for in a solution to help your organization achieve observability.

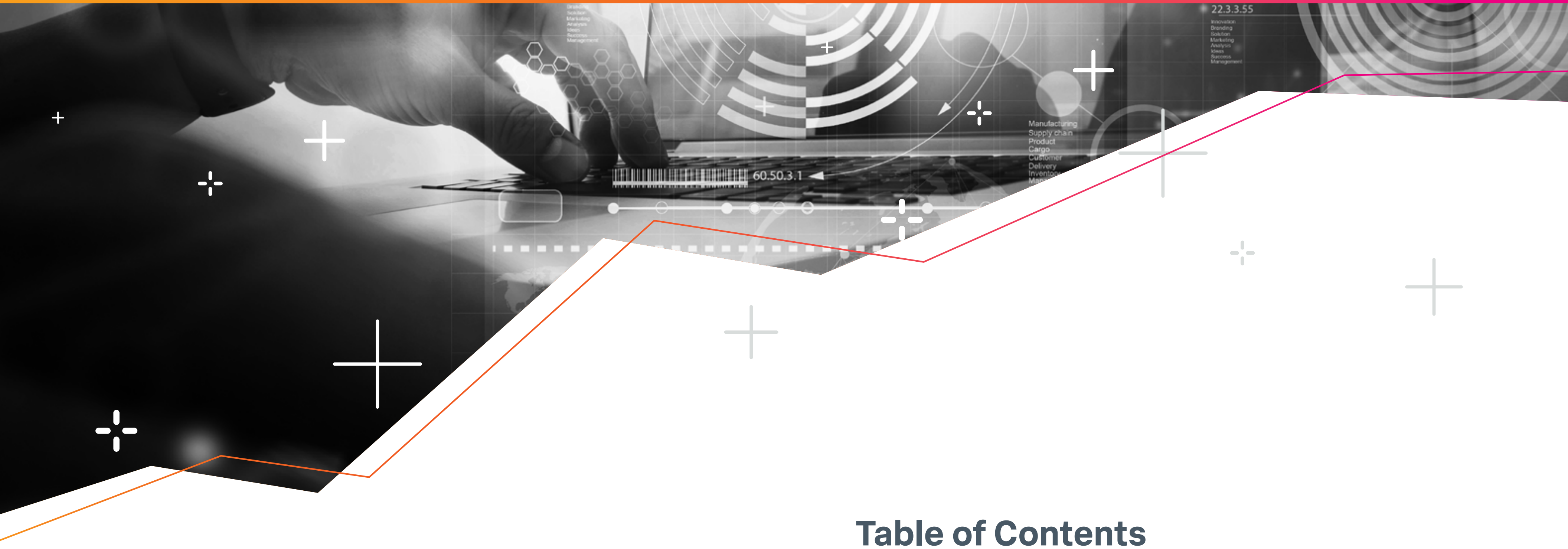


Table of Contents

Visibility
Feedback
Metrics
Instrumentation
Exception Tracking
Controllability
Events
Operational Intelligence
Monitoring
Logs
Externalized State
Questions
AIDev
Tracing
Exhaust
Availability
Observability
Performance
Complexity

CHAPTER 1

Observability — What It Is and Isn't

INTRODUCTION

Building in Feedback

Simply defined, observability is the ability to answer any question about your business or application, at any time, no matter how complex your infrastructure. How you do this in the application development and operations context is simple — by instrumenting systems and applications to collect metrics, traces and logs, and sending all of this data to a system that can store and analyze it and help you gain insights.

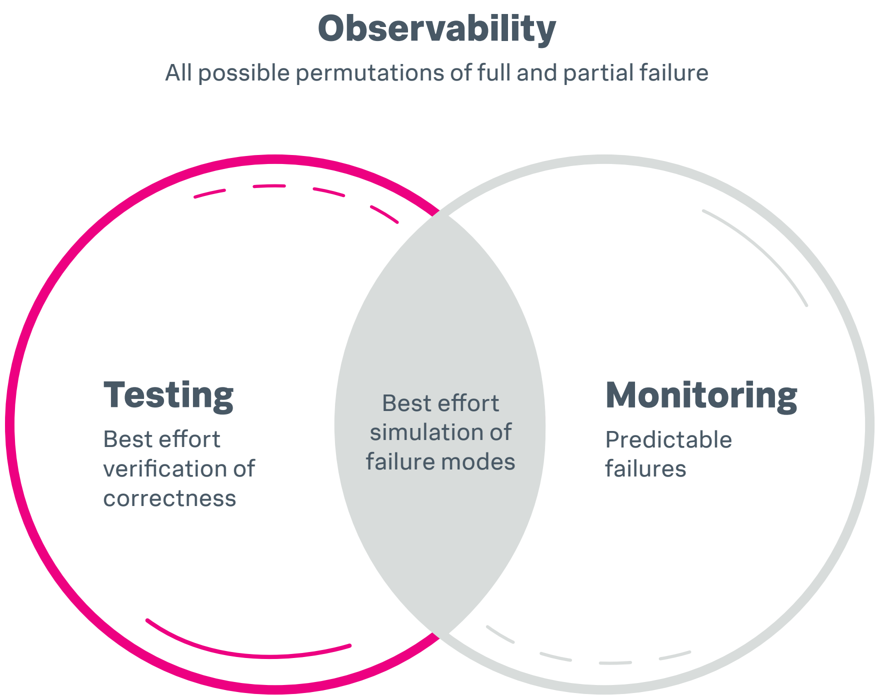
Foundational to observability is building apps with the idea that someone is going to watch them. The classic definition of observability comes from system control theory, where observability is a measure of how well the internal states of a system can be inferred from knowledge of its external outputs — a kind of digital exhaust. Think of it as a property of a system — another attribute, like functionality, performance or testability. You could also think of it as a mindset permeating design decisions: Does my system have the right instrumentation to help me answer questions about its performance?

"Observability is about getting answers to questions that we didn't know we'd have to ask. When I think about observability, I'm thinking from top down and bottom up. It's the actionable insights you collect from your entire system, not just one piece, that tells you the health of your environment."

— **Brent Miller,** Senior Director of Cloud Operations, Quantum Metric

Monitoring vs. Observability

Monitoring	**Observability**
Tells you *whether* the system works	Lets you ask *why* it's not working
The collection of metrics and logs from a system	The useful insights gained from that data
Failure-centric	About overall behavior of the system
Is "**the how**" / something you do	Is "**the process**" / something you have
I *monitor* you	You *make yourself* observable


Observability
All possible permutations of full and partial failure
Testing
Best effort verification of correctness
Best effort simulation of failure modes
Monitoring
Predictable failures

Observability as a mindset

Monitoring is something necessary, but not sufficient, to create observability. To develop effective monitors, you're likely developing an observability mindset already. What's important to note is that there's no one particular tool that will magically give you observability — observability is a mindset, not an outcome.

Observability as a mindset is the degree to which a team or company values the ability to inspect and understand systems, their workload and their behavior. Observability moves beyond looking at the monitors for every single component of the system and looks at the outcomes of the system as a whole.

Benefits of observability

- Comprehensive understanding of complex systems
- Smarter planning for code releases and application capacity
- Faster problem solving and shorter MTTR
- More insightful incident reviews
- Better uptime and performance
- Happier customers and more revenue

Modern infrastructure has evolved from a monitoring mindset to an observability one. In the old days, it was vital to monitor the health of individual services because one service was responsible for most of a user's experience. In modern applications, many services are used to provide the user's experience. If one instance of one service is having a problem, who cares, as long as the user can still do what they need to? After adopting an observability mindset, you're paying attention to the overall system and the user's experience, not each and every component of it. Observability helps you focus on what really matters.

"Observability is not the microscope. It's the clarity of the slide under the microscope."

— Baron Schwartz

CHAPTER 2

Roadmap to Observability

Pillars of Observability

There are three pillars that are needed for observability. More data is always better, but without these, it will be difficult to get the benefits of observability:

Logs/events

Immutable records of discrete events that happen over time

Metrics

Numbers describing a particular process or activity measured over intervals of time

Traces

Data that shows, for each invocation of each downstream service, which instance was called, which method within that instance was invoked, how the request performed, and what the results were

Modern Event-Handling Techniques

Processing the data you've gathered into insights that provide observability facilitates several things, especially shared insights, collaborative response to incidents, data-supported development and intelligent operations. To get these benefits, you need a system that can do these things:

Collect all data

You need to be able to see across stacks, technologies and all environments:

- Cloud-native (containers, cloud, serverless)
- Traditional (Self-hosted, on-premises, monoliths)
- Support for all languages and frameworks you use

Analyze and de-duplicate

- Separate valuable signals from the noise
- Store statistics about your data at ingest time to get to alerts and insights faster
- Detect outliers or other anomalies automatically

Add context

- Show the responding engineer what they need to fix the problem fast
- Minimize downtime by viewing related data to incidents with one click
- Determine the effects of code deployments on key metrics

Role of AI and ML

The volume, velocity and variety of the data needed to answer any question about your business is huge — and fundamentally unmanageable by humans. As buzzword-compliant as it is, to truly get to observability, sophisticated analytic techniques using artificial intelligence (AI) and machine learning (ML) are essential.

High-quality observability systems have learning algorithms that can understand the past health of your services and applications to help predict what's going to happen in the future. Fully ingesting all of the data about your business helps machine learning models get accurate perspectives of historical and real-time data — ML helps to predict high-likelihood, potential future events and harnesses the power of AI to achieve predictive intelligence.

AI-driven analytics

Advances in AI can benefit you by doing the following:

- Reducing event clutter and false positives with multivariate anomaly detection
- Automatically concealing duplicate events to focus on relevant ones and reducing alert storms
- Easily sifting through vast amounts of events by filtering, tagging and sorting
- Enriching and adding context to events to make them informative and actionable

"A learning machine is any device whose actions are influenced by past experience."

—Nils John Nilsson

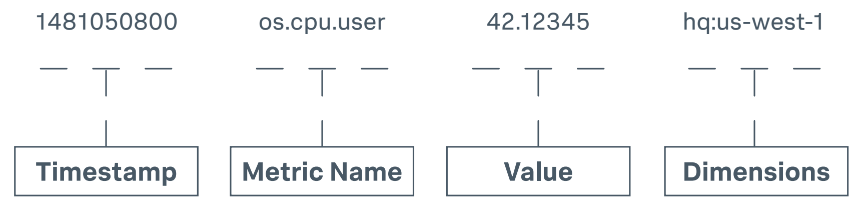
The metrics that matter

Analysts at places including Gartner, Forrester, IDC and Computing UK have all developed their own sets of "metrics that matter." Based on these, the following is a list of metrics and events that we've found to be critical for achieving full observability:

Metrics

Common metrics sources include:

- System metrics (CPU, memory, disk)
- Infrastructure metrics (AWS CloudWatch)
- Web tracking scripts (Google Analytics, Digital Experience Management)
- Application agents/collectors (APM, error tracking)
- Business metrics (revenue, customer sign-ups, bounce rate, cart abandonment)



Events (Logs)

Events come in three forms — plain text, structured and binary. Common event sources include:

- System and server logs (syslog, journald)
- Firewall and intrusion detection system logs
- Social media feeds (Twitter, etc.)
- Application, platform and server logs (log4j, log4net, Apache, MySQL, AWS)

Traces

Specific parts of a user's journey are collected into traces, showing which services were invoked, which containers/hosts/instances they were running on, and what the results of each call were.

Collecting observability and monitoring data

The good news is that so much data exists; the challenge is aggregating and gaining insight from all of it. The following are types of data sources that have evolved over the years — all important in achieving observability.

Existing sources

- Network flow data: router/switch counters, firewall logs, etc.
- Virtual servers: VM Logs, ESXi Logs, etc.
- Cloud services: AWS data sources such as EC2, EMR, S3, etc.
- Docker: logging driver, syslog, apps logs, etc.
- Containers and microservice architectures: container and microservices logs, container metrics and events, etc.
- Third-party services: SaaS, FaaS, serverless, etc.
- Control systems: vCenter, Swarm, Kubernetes, etc.
- Dev automation: Jenkins, Sonarcube, etc.
- Infra orchestration: Chef, Puppet, Ansible, etc.
- Signals from mobile devices: product adoption, users and clients, feature adoption, etc.
- Metrics for business analytics: app data, HTTP events, SFA/CRM
- Signals from social sentiment analytics: analyzing tweets over time
- Customer experience analytics: app logs, business process logs, call detail records, etc.
- Message buses and middleware

Cloud-native sources

State-of-the-art

- OpenTelemetry: a framework that integrates with most OSS and commercial products to collect metrics and traces from apps written in many languages

Used by early cloud adopters

- collectd: a daemon that collects metrics
- statsd: a daemon that listens for statistics
- fluentd: a daemon that unifies log data collection
- Zipkin, Jaeger: Open source back-end distributed tracing systems

These daemons send metrics to a defined location vs. observability, which creates and defines the metrics that matter and will drive action when those metrics are out of limits.

It's important to note that no code is "done" until you've built analytics and instrumentation to support it. This is especially important to remember as you pursue an observability mindset.

Without building in this kind of visibility, you're unable to determine *why* a system has failed, which slows response and resolution time for business-critical issues.

Proper instrumentation is vital to getting the meaningful results you're looking for in an observability system in the first place. Adopting OpenTelemetry can provide the fastest path to having instrumentation done and getting the benefits of observability sooner.

Tools needed for observability

There are many solutions that can help you get insights from the overwhelming volume of disparate data from all the sources listed above, but you'll likely find that you need the following tools to answer all the questions about your application, and to gain a full, end-to-end picture of it. It's important to remember that you want the fullest fidelity possible in all of these tools:

Tool	Use
Infrastructure monitoring	Determine the health and performance of the containers and environment your applications run on.
Application performance monitoring	Investigate the behavior of your application at the service level. Determine where calls are going and how they perform.
Real user monitoring	Understand the experience of real users by collecting data from browsers about how your site performs and looks. Isolate issues to the frontend or backend.
Synthetic monitoring	Measure the impact that releases, third-party APIs and network issues have on the performance and reliability of your app.
Log viewing	Dig deeper into "the why behind the what" when issues occur. Figure out how to remediate the issues quickly.
Incident response	Alert the right team the first time to fix the issue and provide them with the data they need to succeed in doing so, all in one place.

CHAPTER 3

Observability in Action

INTRODUCTION

Now that we've talked about what observability is, why it's important, and the metrics and events it's based on, let's see observability in action.

The following case studies present real customer data and results from organizations using Splunk's Observability Cloud products.

Application Performance Monitoring

Quantum Metric – Rapidly-growing unicorn Quantum Metric needed a flexible observability solution that collected all the data about their environment in one place. Businesses around the world rely on Quantum Metric to power real-time customer insights, such as when their application is experiencing errors or drops in conversion rates. Adopting Splunk Observability Cloud helped Quantum Metric get better understanding of their complex infrastructure and also provided the following benefits:

- Better downsizing analysis and capacity planning saved over $80,000.
- Application development was made 96% faster, increasing developer productivity.
- The number of pending CI jobs was reduced by 95%.
- Customer-specific service level objectives (SLOs) with built-in detectors alert them immediately when issues arise.

"Observability is about getting answers to questions that we didn't know we'd have to ask."

—**Eric Irwin,** Director of Engineering

Splunk Application Performance Monitoring

Freecharge — India's leading digital payments app was running into obstacles while trying to migrate to a microservices-based environment. Previously, multiple monitoring tools were needed to get visibility into their complex application. They adopted Splunk Application Performance Monitoring and saw the following benefits:

- KPI monitoring helped the customer success team resolve issues with uptime and latency 60% faster.
- Over a thousand compute instances were reduced to under 700, resulting in significant cost savings.
- The business team is alerted to issues in minutes, compared to hours with prior tools.

"[Splunk Infrastructure Monitoring] monitors the heartbeat of the entire system our customers interact with — not just our internal infrastructure, but also the external providers we're integrated with via API — so we can easily pinpoint and resolve issues regardless of where they are in our ecosystem."

— **Sachin Sharma,** Senior Director, Infrastructure

Splunk Infrastructure Monitoring

Mark43 — Mark43's prior infrastructure monitoring platform was unstable and noisy, increasing workload and burying legitimate issues. After adopting Splunk Infrastructure Monitoring, Mark43 now:

- Is alerted to issues in seconds, compared to 10-minute latency with previous tools.
- Sees improvements (which previously took a full team almost an hour) done by one engineer in minutes.
- Can fix issues before they have a downstream effect for the officers using the platform.

"Performance improvements that previously took a full team almost an hour can be done by one engineer in minutes."

— **Kevin Heins,** DevOps Technical Lead

Splunk Infrastructure Monitoring

Acquia — Digital-experience platform Acquia was undergoing explosive growth and quickly outgrew their homegrown monitoring system. Modifications took weeks, and data was missing or incomplete throughout the system. Reinvesting efforts into their core competency, Acquia adopted Splunk Infrastructure Monitoring and realized these benefits, in addition to gaining full observability into more than 20,000 EC2 instances:

- Ability to send and analyze 300 metrics with 4x more granularity than before.
- $1 million in annual productivity gains from 26% less time spent per incident.
- $600,000+ annual AWS infrastructure savings.
- Time savings of an hour per day per technical employee.

“Splunk allows us to accelerate product development, improve support efficiency and provide business critical analytics to our customers.”

— **Aaron Pacheco,** Product Manager

Splunk Observability Cloud

Atlassian — Industry-leading cloud technology provider Atlassian moved from a competing product to Splunk Observability Cloud because the competing product couldn't operate at Atlassian's scale. They were encountering slow performance, unclear billing and a lack of transparency around usage. Atlassian moved to Splunk Observability Cloud, which now supports:

- Delivery of 1,000+ high-reliability services across 150,000+ customers.
- A massive monitoring infrastructure that handles 1.5 million+ metrics/sec and 1,000+ dashboards and detectors.
- A huge team of 2,000+ developers and SREs.
- A standard, Terraform-based approach for "observability as code."

Splunk Observability Cloud

Rappi — Number one Latin American e-commerce company Rappi's hockey-stick growth, combined with the adoption of containers and microservices across 6,000+ hosts, strained their legacy monitoring platform, which lacked sophisticated and granular analytics, resulting in long delays to deliver alerts. After adopting Splunk Observability Cloud, Rappi:

- Gained real-time observability across their environment.
- Reduced the MTTR in production from five minutes to seconds.
- Accessed more complex data analytics and better metrics correlation, reducing MTTR.

Grew confident in their continued migration to a microservices and serverless architecture, including ECS, Kubernetes and AWS Lambda (100+ services).

Splunk Incident Response

PSCU — Financial technology company PSCU knows that downtime costs money, and they were experiencing MTTAs of approximately four hours before adopting Splunk On-Call. Before using Splunk On-Call, a simple round-robin escalation process existed. After one year of using Splunk On-Call, their MTTA had dropped an astounding 90% to 20 minutes. Further use of Splunk On-Call dropped their MTTA even further. They also saw:

- On-call schedules managed across departments in one tool for maximum visibility.
- Increased accountability for on-call teams with tracking of incidents and their outcomes.
- Improved on-call experience because alerts went to the right person the first time.

"With Splunk On-Call, we took our MTTA from four hours to two minutes and achieved stronger call-team accountability."

— **Earl Diem,** IT Operations Manager

CHAPTER 4

Options in Observability

INTRODUCTION

As the need and demand for observability grows, some monitoring tool vendors are, of course, jumping on the bandwagon about as fast as they did with DevOps a while back.

However, it's important to note that no tool is going to "give you" observability. Observability is a mindset — not a practice — so any vendor promising you 'observability in a box' should raise red flags as you research your options.

Additionally, many vendors claim to have full observability capabilities, but upon closer inspection, they only offer a portion of the observability picture. If you're going to spend the effort to adopt an observability mindset, you need to make sure that your tool can handle all of the data and provide you with real insights.

Observability Built for Complexity

Modern development and deployment practices create increased complexity, as we've discussed. This complexity makes it more and more difficult to operate applications reliably. The end goal of an observability tool should be to make your insights more useful and your business more effective. We recommend that you consider to what extent any tool you evaluate helps you do better at these three key operational functions:

Monitor

to get centralized visibility into all relevant infrastructure and applications, no matter where or how they are hosted. Look into containers and across cloud and hybrid environments. Detect root cause of issues and provide context. Generate automated insights to detect problems early.

Collaborate

across organization silos and share the data that matters with every stakeholder to manage incident resolution and build repeatable action plans. Make sure that the correct person is engaged for any alerts the first time. Share views into issues or anomalies that don't require explaining how to re-create them.

Automate

the mundane, orchestrate the complex. Automate processes to save time to focus on what really matters — like strategic business initiatives — rather than just monitoring and troubleshooting. Eventually, orchestrate multiple processes to optimize (and minimize) human intervention in monitoring, root cause analysis and remediation.

Why You Need Splunk Observability Cloud

Your infrastructure and applications generate huge amounts of data every second. Capturing all of that data and using advanced analytic and processing techniques helps you conquer complexity, solve problems faster, improve user experience, and take advantage of all your data how you want to with OpenTelemetry. Splunk Observability Cloud bundles all the necessary components to help you build your observability mindset: infrastructure monitoring, application performance monitoring, real user monitoring, synthetic monitoring, log exploration and incident response.

Splunk Observability Cloud's most important benefit is helping make sense of the complex architecture created by modern development practices. Even if your organization hasn't embraced a cloud-first solution, new development will slowly move in that direction over time. Containerization, public or private cloud hosting, serverless functions and the like all accelerate development time and make innovation easier, but they also make getting a sense of what's going on more difficult. Splunk Observability Cloud consolidates data across any hosting environment, for any application, and is the only tool that can accept *all* your data, making sure you don't miss anything.

Issues happen with every system. No matter what, things will go wrong and it's important that they can be found and fixed quickly when they do. Many organizations currently just perform reactive troubleshooting — they find out something is wrong, and then start looking through log files to find the cause. With Splunk Observability Cloud, you can see what went wrong and where for a particular user in a few clicks, rather than having to bounce all over your infrastructure trying to find the problem.

Ultimately, all applications are developed to provide a service to users. Observability systems that don't take real user experience into account don't provide a full picture of what's happening. Splunk Observability Cloud provides a way for you to look at the actual, real-world experience any user has by seeing exactly how long each part of your page load took and even recommending ways to improve that performance.

Finally, instrumenting all your applications and infrastructure is a time-consuming process. It's the kind of thing that you need to do, but that ideally you should only do once. Splunk Observability Cloud was built for OpenTelemetry, an industry-standard instrumentation system. OpenTelemetry is the second most active Cloud Native Computing Foundation project and other open source and commercial projects are increasingly adopting it. Instrumenting your applications with OpenTelemetry means you only have to instrument once and can then take your data to any other provider in the future. As more projects adopt OpenTelemetry, you may even find that new applications come pre-instrumented and ready to emit data to Splunk Observability Cloud out of the box.

Observability:

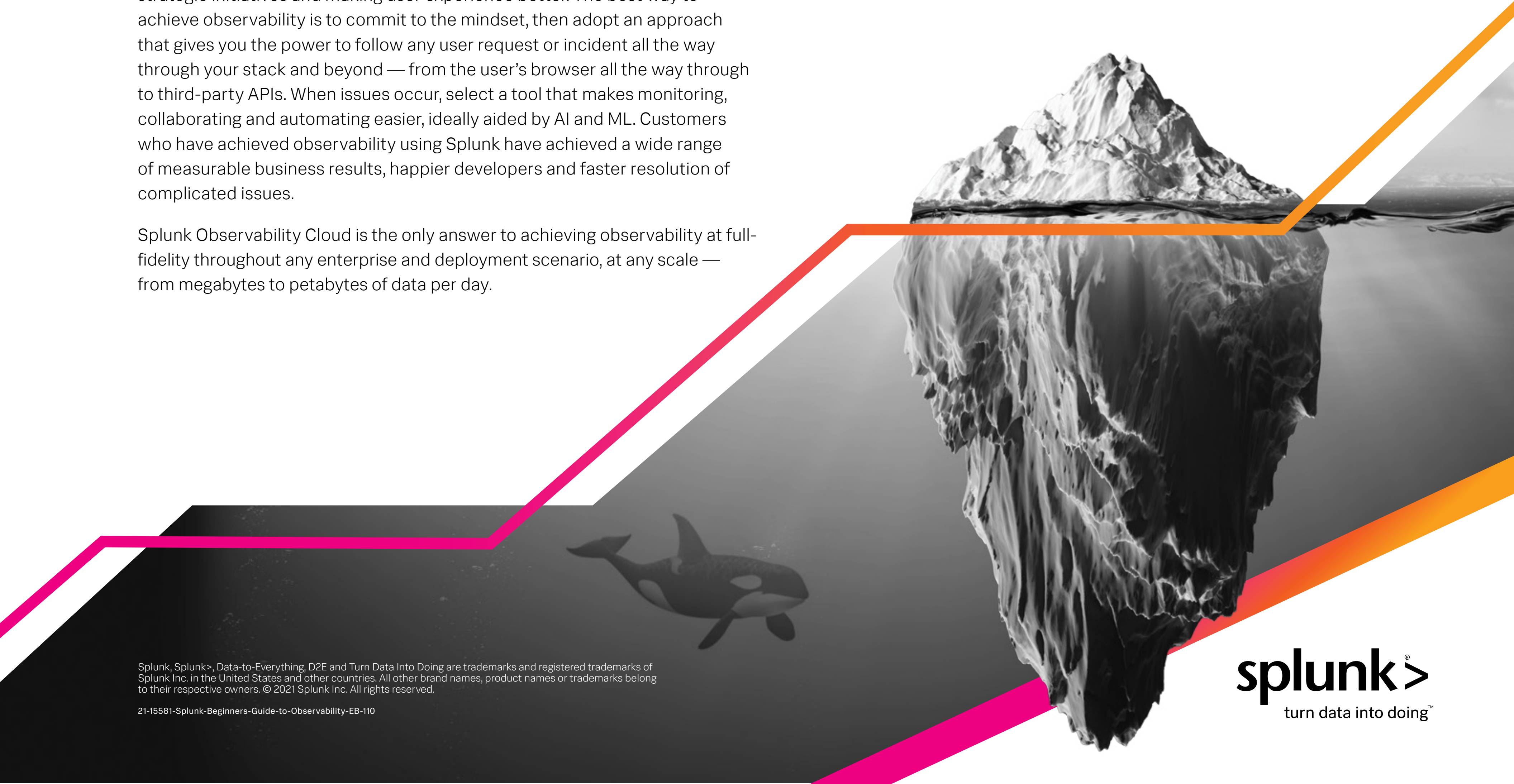
- Lets you conquer the complexity of modern architecture and applications and answer any question about your application and business.
- Is a mindset, not a practice.
- Absorbs and extends classic monitoring systems to let you answer questions, not just hear that something is wrong.
- Makes use of all your data to automatically provide insights, predict errors, and make apps better and users happier.
- Helps you solve problems in seconds, not hours. Predicts what could go wrong and helps identify the root cause of issues.
- Gives you the power to continually improve user experience by seeing snapshots of exactly what's happened for each of your users.
- Allows you to have full control over your data and to leverage the open source community to accelerate your observability journey.

Conclusion

The hype of observability is well earned. It allows engineering teams to take greater ownership of uptime and performance, and it requires an organizational shift to succeed. Observability cuts through the complexity of modern architecture and provides end-to-end visibility across your system so you have outcomes that are quantifiable.

You can quickly fix and eventually prevent problems, leaving more time for strategic initiatives and making user experience better. The best way to achieve observability is to commit to the mindset, then adopt an approach that gives you the power to follow any user request or incident all the way through your stack and beyond — from the user's browser all the way through to third-party APIs. When issues occur, select a tool that makes monitoring, collaborating and automating easier, ideally aided by AI and ML. Customers who have achieved observability using Splunk have achieved a wide range of measurable business results, happier developers and faster resolution of complicated issues.

Splunk Observability Cloud is the only answer to achieving observability at full-fidelity throughout any enterprise and deployment scenario, at any scale — from megabytes to petabytes of data per day.



splunk>
turn data into doing™