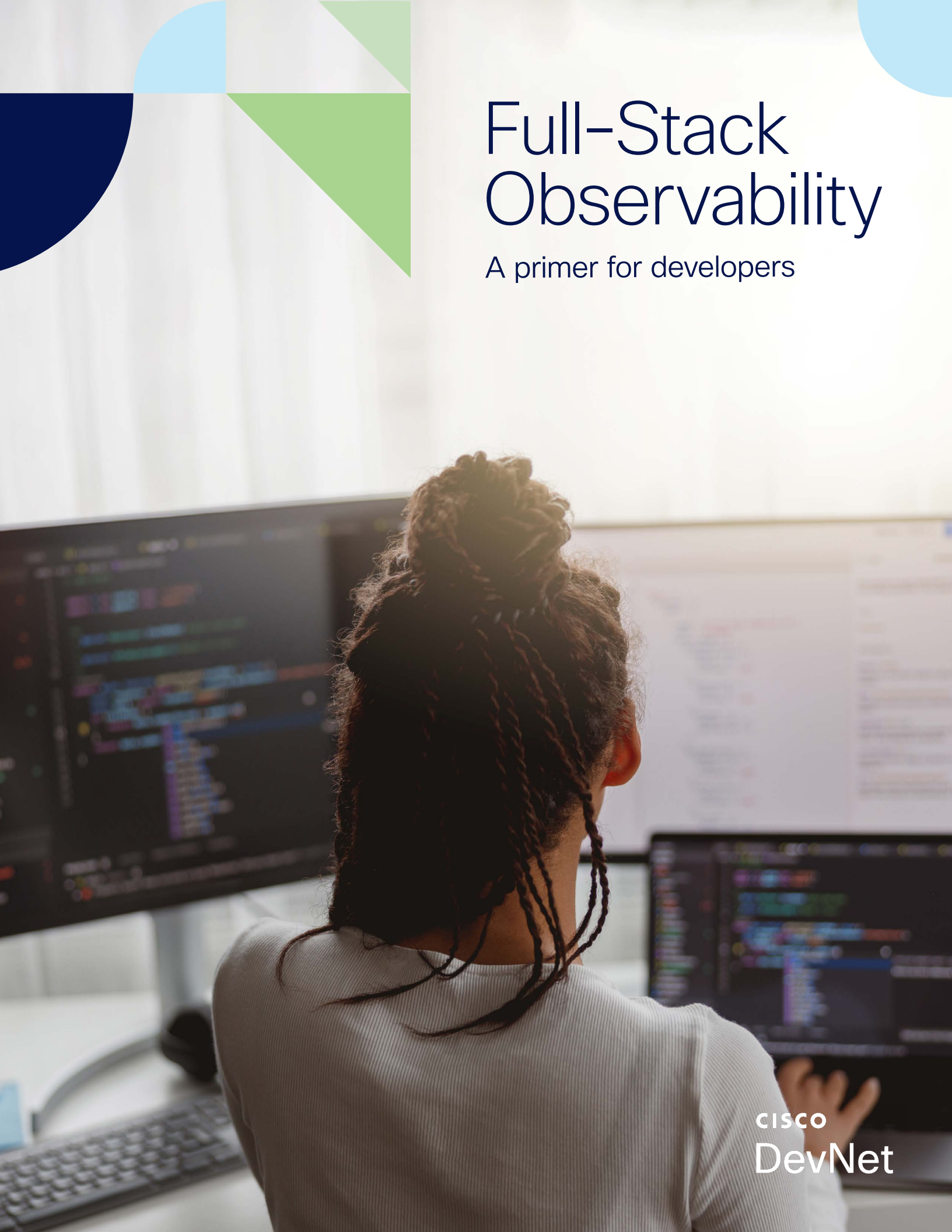
Full-Stack Observability

A primer for developers



cisco DevNet

Developers and the demands of today's business

The face of business is now online. Applications are today's marketplace ambassadors, positioning technology front and center in the customer conversation. Developers, with greater responsibility for shaping a company's commercial identity, are working faster, building smarter, and creating more value in an effort to keep pace with shifting competitive and operational demands.

Technology has evolved to meet these imperatives and, in the process, grown far more complex. New components, new endpoints, new environments – all interconnected in dynamic, disparate on premises and cloud ecosystems sometimes too tangled for humans to understand, especially those whose focus is on shipping clean code, fixing problems in real time, and contributing to the integrity, security, and performance of the systems running their applications.

Amid all this complexity, **full-stack observability** (FSO) is a powerful framework for delivering insights and supporting developer agility and decision-making. An evolution and expansion of application performance monitoring, FSO offers a way to see inside systems and their components, across the technology stack - be they networks, a virtual server, or a Kubernetes cluster. FSO can surface information about components and their relationships to help developers create, launch, and manage applications more effectively.

That's important since applications now depend on so many services to do their work. Research firm IDC found that applications rely on anywhere from 5 to 15 separate services, resources, or APIs, on average, each with its own operational risks.

Understanding these interdependencies and risks across technology layers is key for developers and operators as they fix bugs, improve code, find infrastructure efficiencies, secure the components running applications, and improve customer experience. When it comes to working faster and smarter, developers can lean on FSO outputs to achieve their goals with less operational friction. Questions like "how do I fix this?" or "what affect will my new code have on the application?" are easier to answer with information and insight into system functioning made possible by FSO.

What distinguishes the FSO approach is its focus on correlation, insight, and proactivity over silos, alerts, and responses. FSO builds bridges by nurturing a common understanding of systems from data, promoting shared responsibility and problem-solving across ops and development teams, grounded in deeper levels of system knowledge. It also pulls security into this sphere of oversight.

When aligned to business outcomes, as FSO should be, it can help developers answer questions about commercial priorities and customer needs. It can also deliver insights for optimizing infrastructure costs and performance. Observability data can help fine-tune applications earlier in the development cycle and for assist teams in locking in on potential security vulnerabilities.

This e-book is an introduction to the building blocks of FSO and the potential benefits it can bring to developers' everyday work. It describes the four key data types used by FSO, and explains why traces are critical for cross-domain visibility. It covers instrumentation and the important role OpenTelemetry can play in building effective FSO solutions. Finally, it explains key use cases for FSO in the enterprise, and how developers can use it to change business, organizational, and operational outcomes.

Data: The foundations of FSO

The foundations of full-stack observability are built on data. The four types of data used in observability solutions are metrics, events, logs, and traces (MELT). Each type of MELT data contributes specific value to the visibility and insight puzzle. Together, they're the raw materials for gaining a deeper understanding of systems.

The power of the data is in their correlation across domains, since applications depend on constellations of connected components to run. Creating a single source of the truth with the data can help tear down traditional operations silos that can create provincial, domain-specific thinking rather than the holistic, cross-team customer and business vision necessary for optimal system engagement.

While all important to system visibility, traces advance the cause of observability by revealing the chain of actions an application takes to complete a task. Spans represent a single operation within a trace Given the complexity of today's system architecture, that could include calling an API to execute functionality inside a cloud-based container or pull proprietary data data from an on-prem server. Traces track the connections between these actions and locations. That benefits developers in a number of ways. First, in the development lifecycle, understanding these connections can help pinpoint issues within the stack and target the areas that need de-bugging and improvement. With all the pressure developers are under to release more code more quickly, shipping flawed code is unavoidable. Finding problems earlier in the development lifecycle can limit the mistakes that go to production.

Observability

To understand how the elements of MELT can help, it's useful to define each:

M

Metrics

Numeric measurements grouped or collected at regular intervals or over a given time span.

E

Events

A discrete action happening at a moment in time. Abstracted yet still highly detailed, these are critical for observability.

L

Logs

Strings of text with an associated timestamp describing a system action. The original date type.

T

Traces

Chains of event between different components in an application.

cisco DevNet

> When it comes to working faster and smarter, developers can lean on FSO outputs to achieve their goals with less operational friction.

In remediation, meanwhile, traces can speed up root cause analysis and lead to faster issue resolutions. When optimizing systems, they can inform coding and infrastructure decisions, provide evidence for shaping customer experience, and help to identify ways to trim operating costs.

Events, for their part, are the flags indicating that specific changes have occurred. These might be undesirable actions that require attention,w positive actions that confirm the system is running optimally, or everyday actions that reflect that the system is performing the work that it was intended to do.

Metrics are aggregated raw data, can be measured against KPIs, and trigger events on system anomalies. Generally, they're less costly to store and process since they're data aggregates rather than individual records.

Through logs you can identify root causes and pinpoint failures around an event, troubleshoot, and answer questions about access activities. Historically, developers have relied on logs for clues to understanding the behavior of components and applications.

When artfully combined, these data types establish a fundamental understanding of the connections throughout and the health and performance of systems. For developers, they help answer pressing questions and facilitate day-to-day work. That's observability.

cisco DevNet

Seeing into systems

To generate all this data, a system needs instrumentation. In FSO, that instrumentation ideally sits on every system component so every part of the IT environment is observable. After all, you can't fully understand what you can't see.

The boundaries of today's IT, though, are shifting, often on and off premises, and are sometimes ephemeral. The public cloud, virtualization, remote devices, and containerization are just some of the features of modern system architecture that challenge visibility and broad system oversight. Old monitoring techniques are hard to implement when parts of your infrastructure are owned by someone else or move around asynchronously.

Instrumentation, or telemetry, has had to evolve. That's especially true since an application might leave its footprint on so many different components as it performs the steps of a request/response chain. Connecting and seeing those steps is key to understanding the stresses, choke points, and excess cycles that prevent applications from running optimally for the business and

Standardization is key. **OpenTelemetry (OTel)** is emerging as the open-source industry standard for instrumentation, data collection, and delivery into many common observability back-ends. OTel collects, collates, and sends telemetry data in a consistent and flexible way, no matter where you're installing the instrumentation. It can span components on-premises, in the public cloud, and at the edge.

That's important since true full-stack observability is only possible with comprehensive telemetry and data that illuminate interdependencies and the state of systems at any given time.

What makes OTel a solution fit for FSO is the fact that it is:

Standardized: It offers a single, vendor-agnostic instrumentation library, a vendor-neutral collector, and SDKs in multiple languages, removing the need to manage multiple libraries or formats.

Controllable: It has the flexibility to send to multiple back-end observability platforms. It can receive data in one format and deliver it in another. It can also help manage data overload.

Portable: It separates data collection and delivery from the tools that ingest and analyze it. It eliminates the need to install proprietary or manual data collection libraries. The data pipe can be unplugged from one back end and plugged into another, which frees solutions from vendor lock-in.

Supported: It's open source and widely adopted. Many developers, including those at Cisco, contribute to its development.

From data to insight to action

FSO moves the developer narrative from the “what” and “when” to the “why” of system understanding. It is the necessary next step to navigating IT complexity and controlling resources across domains. The goal is to turn the visibility that data provides into insight and then insight into action, focusing on three high-level areas, which are interdependent and underpinned by business context.

Optimization

The focus here is on cost and efficiency. As systems get more complex, they get harder to visualize and understand, both from a logical standpoint and in terms of utilization. Everything from setting up microservices to provisioning infrastructure is harder without insight into how each component is working. It’s not unusual for developers to request redundant resources to run applications or to overestimate needed capacity when visibility into system architecture is challenging. Through full-stack observability, developers can get a better handle on resources to optimize utilization, remove redundancies, and better manage infrastructure costs.

Performance

Performance touches developers and customers alike. For developers, an observable system is an easier system to work with. It reveals problem areas in code so developers can isolate bugs early in the development cycle and fix them before release. Developers spend a significant amount of time resolving problems caused by code changes, and an observable system can cut down on the time understanding the impacts. Developers also fix runtime issues, and an observable system can shorten mean time to resolution, again, by providing a system roadmap to trouble spots.

Security

Observability and security are beginning to converge as organizations move from a monitor-and-react model of system management to a paradigm of surveying, interpreting and acting. With the pressure on developers to continuously release code into increasingly complex IT environments, maintaining software reliability is more and more dependent on teams’ ability to spot and close vulnerabilities. Observability, alongside automation, can play a crucial role in risk management within the mix of security solutions companies employ.

Business Context

So much of today’s IT architecture is business architecture. So, aligning SLAs through to the SLOs and SLIs of technology is really about ensuring systems are delivering on business objectives and that teams share a common business context, which was never really possible before. That’s an important function of full-stack observability – to track and measure system behavior to ensure that customers are getting what they expect from an optimized, performant, and secure technology stack.

cisco DevNet

Helping developers to develop

Developers are in high demand. There are high expectations on them to deliver. One way to manage heavy workloads and prevent burnout is to find ways to make the job easier. Observability tools for developers aren't so much about understanding the state of systems in real time; they're about supporting their day-to-day workflows and removing barriers that can slow delivery and innovation.

Observability tools can answer questions developers face regularly, like "how do I fix this?" But they can also help with decisions about adding or deprecating application features, how to scale applications, and the impact of changing out components. Cloud dependency creates challenges in system transparency, and observability tools can bring visibility across environments to make it easier to understand system functions when introducing changes to source code.

This improves productivity and agility. It also enables developers to be more responsive to issues since they'll spend less time figuring out what happened and more time answering "why" so it won't happen again.

FSO also makes tighter allies of ops and dev teams as it unites them around a common IT-to-client context. This aids in issue identification and problem-solving, as it gets everyone pulling together toward a shared goal.

Developers are beginning to embrace the possibilities of observability for continuous feedback, learning opportunities, and this operational insight.

There are, of course, benefits to customers. Fewer bugs, more uptime, faster response times to issues, tighter security, and greater alignment to business needs all support a better application experience, which today is pivotal to commercial success.

To summarize some key ways FSO can support developers:

- Improve productivity and agility
- Better align dev teams with ops
- Deliver deeper business context to aid decision-making
- Enhance responsiveness
- Provide broader visibility into system DNA

Owning the environments

As business continues to go digital, organizations are asking for more velocity, agility, and precision output from developers. Customers, more and more, expect their vendors to get the online experience right the first time, leaving less room for buggy code or sub-optimal features in production and a smaller margin of error for unexpected application downtime. In the midst of this, enterprises are moving to or exploring more complex multi-cloud deployments, and need the ability to break up legacy applications and become micro-services based.

To respond, developers are connecting more with systems throughout the development life cycle and taking more ownership of the environments that run their applications, from pre-production to production. They're partnering more closely with operations teams to speed up problem solving and optimize code and infrastructure. The goal is to release cleaner, more efficient code, to reduce issues and vulnerabilities, and to deliver the best online experience for customers.

FSO offers opportunities for developers to deepen their engagement with and understanding of systems and to apply meaningful insights that help them work smarter, faster, and toward a set of business objectives shared by all.

Discover MELT data

Get started with OpenTelemetry

Visit the FSO Developer Hub