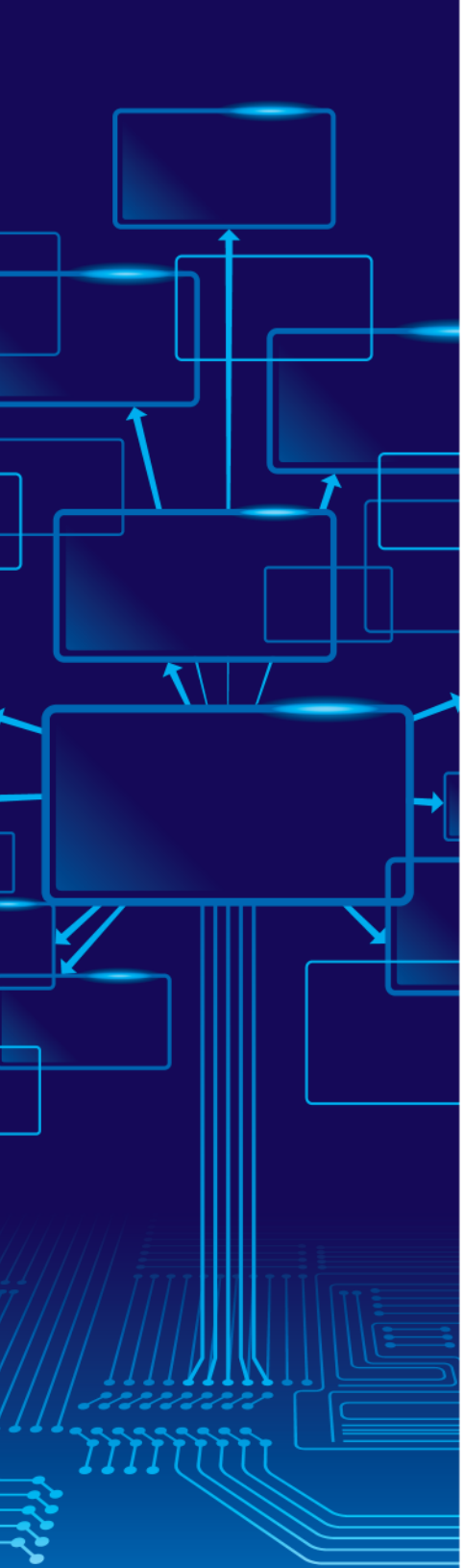


What is observability? A beginner's guide

TechTarget

In this guide:

Observability is an extension of traditional IT monitoring; it uses machine learning to broaden the range of monitoring oversight to cover not only logs and traces or network traffic and storage devices, but social media feeds and other custom non-technical data feeds, as well. This increase in comprehensiveness offers IT teams a much more dynamic view into their overall ecosystem, which enables unique insights IT monitoring alone can't provide.

Machine learning acts as a backbone to observability, enabling a tool to connect the presence of an alert -- high RAM use, for example -- to the reason for its occurrence -- all via a single dashboard -- rather than leave IT admins to conduct that search mission manually. Observability tools present IT admins with the 'what happened' and 'here's why' components of an alert simultaneously before the alert occurs, which facilitates proactive management, as opposed to reactive. When incorporated into a cohesive DevOps [people] environment, it enables IT staff to identify issues before they even cause a problem.

This Tech Accelerator package covers what observability is, why it’s important, what the benefits and challenges are and how to implement it.

In this guide:

What is observability? A beginner's guide

STEPHEN BIGELOW, SENIOR TECHNOLOGY EDITOR

Observability is a management strategy focused on keeping the most relevant, important and core issues at or near the top of an operations process flow. The term is also used to describe software processes that facilitate the separation of critical information from routine information. It can also refer to the extraction and processing of critical information at the highest-level architecture of operations systems.

Observability is an element in control theory, which says that the internal states of IT systems can be deduced from the relationship between their inputs and outputs. Thus, it is also often described as a top-down assessment. The challenge of observability lies less in deriving the internal state from observations than in collecting the right observations.

WHAT ARE THE DIFFERENCES BETWEEN MONITORING AND OBSERVABILITY?

The concepts of monitoring and observability are related, but the relationship is complex. The following are some of the major differences:

- Monitoring tools passively gather information, most of which turns out to be insignificant. This can drown operations personnel and even AI tools in data.

In this guide:

Observability actively gathers data to focus on what's relevant, such as the factors that drive operations decisions and actions.

- Monitoring tends to gather information from available sources, such as management information bases, application programming interfaces (APIs) and logs. While observability will also use these sources, it will often add specific new points of information access to gather essential information.
- Monitoring focuses on infrastructure, where observability focuses equally on applications. That means observability will often include a focus on workflows, whereas monitoring focuses on point observations.
- The data made available through monitoring is often the sole expected outcome. Observability presumes that data sources will contribute to an analytic process that will then represent the state of an application or system optimally.

WHY IS OBSERVABILITY IMPORTANT?

For decades, businesses that control and depend on complex distributed systems have struggled to deal with problems whose symptoms are often buried in floods of irrelevant data or those that show high-level symptoms of underlying issues. The science of root cause analysis grew out of this problem, as did the current focus on observability. By focusing on the states of a system rather than on the state of the elements of the system, observability provides a better view of the system's functionality and ability to serve its mission. It also provides an optimum user and customer experience.

In this guide:

Observability is proactive where necessary, meaning it includes techniques to add visibility to areas where it might be lacking. In addition, it is reactive in that it prioritizes existing critical data.

Observability can also tie raw data back to more useful "state of IT" measures, such as key performance indicators (KPIs), which are effectively a summation of conditions to represent broad user experience and satisfaction.

WHAT ARE THE COMPONENTS OF VISIBILITY?

Visibility is the ability to peel back the layers of a system or infrastructure to gather useful information or perspective. Monitoring tools and associated practices play a vital role in this, providing the scope and depth needed to gather health, security, hardware, host OS and other data needed to gain visibility. The choice of tools and the specific data gathered depend on monitoring goals and the ways that data will be used in observability. Thus, visibility involves tools and policies:

- Tools are the software applications used to gather and aggregate data. This can involve a broad range of native vendor or third-party tools, as well as single-point or multipoint tools. The tools can gather data from systems, networks, applications and platforms, such as hypervisors.

- Policies guide and focus the data gathering needed for visibility and define how data is secured and retained. Policies help prevent an organization from gathering everything and keeping it forever, which can turn an ocean of data for visibility and analytics into an unmanageable swamp.

Remember that more isn't always better. Visibility is about seeing what's important, not seeing everything.

WHAT ARE THE THREE PILLARS OF OBSERVABILITY?

The three primary source data types for observability -- also called the three pillars of observability -- are logs, metrics and traces:

- **Logs.** Records of events, typically in textual or human-readable form, are known as *logs*. They are almost always generated by infrastructure elements, including both network devices and servers. They can also be generated by platform software, including operating systems and middleware. Some applications will log what the developer believes represents critical information. Log information tends to be historic or retrospective and is often used to establish context in operations management. However, there are logs that represent collections of events or telemetry data, and the detailed information can be available in real time.
- **Metrics.** This type of real-time operating data is typically accessed either through an API using a pull or polling strategy, or as a generated event or telemetry -- a push or

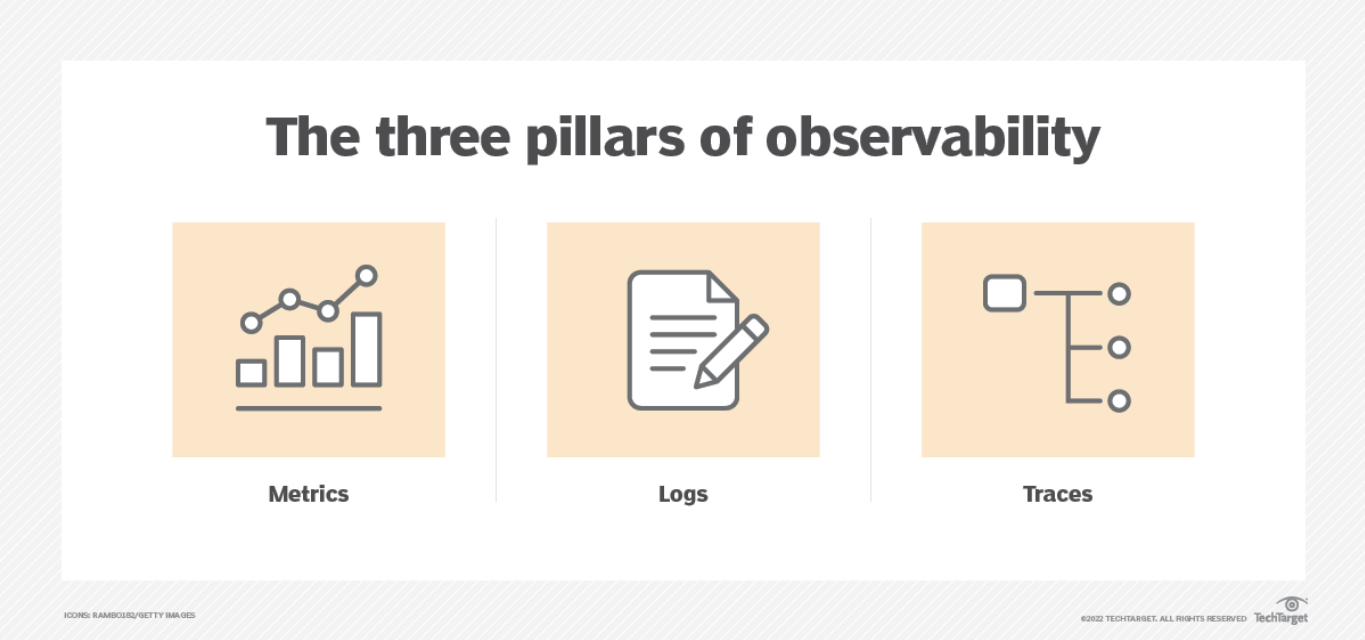
In this guide:

notification, for example. Because they are event-driven, most fault management tasks are driven from metrics.

- **Traces.** These are records of information pathways or workflows designed to follow a unit of work, such as a transaction, through the sequence of processes that application logic directs it to follow. Because work steering is normally a function of the logic of the individual components, or of steering tools like service buses or meshes, a trace is an indirect way of assessing the logic of an application. Some trace data might be available from workflow processes, such as service buses or cloud-native microservices and service meshing. However, it might be necessary to incorporate trace tools into the software development process to gain full visibility.

All three pillars are vital to observability, but each has unique limitations that should be considered. For example, metrics are hard to tag and sort, and they can be difficult to use for troubleshooting; logs can be challenging to sort and aggregate to draw meaningful conclusions or relationships; traces can produce enormous amounts of unnecessary data. Thus, observability practitioners can still encounter limitations in gathering real insight, find far too many places to look for problems or have difficulty drilling down to translate issues into actionable problems.

In this guide:



Practitioners might find it more effective to use the three pillars of observability through a goals-oriented lens: Set business objectives, such as service-level objectives, and then set observability goals that align with those objectives. For example, if the business is concerned with latency or throughput, set appropriate latency or throughput objectives, and then use the three pillars to approach observability with those goals in mind.

WHAT ARE THE BENEFITS OF OBSERVABILITY?

The primary benefit of observability is improvement to the user experience, created by focusing operations tasks on issues that threaten that experience. Proper application of observability can improve application availability and performance.

Observability practices will also normally reduce operations costs by speeding up the handling of adverse conditions. This happens by reducing the amount of irrelevant or redundant information and prioritizing the notification of critical events. These improvements are most noticeable in larger enterprise operations where large operations teams are required.

Some users report that observability practices provide information that is helpful in reliability and performance management, and even in infrastructure design and tool selection. This is because a focus on truly critical information helps identify vulnerabilities that can be corrected by changing configurations, application design and resource levels.

WHAT ARE THE CHALLENGES OF OBSERVABILITY?

Observability does come with challenges, including the following:

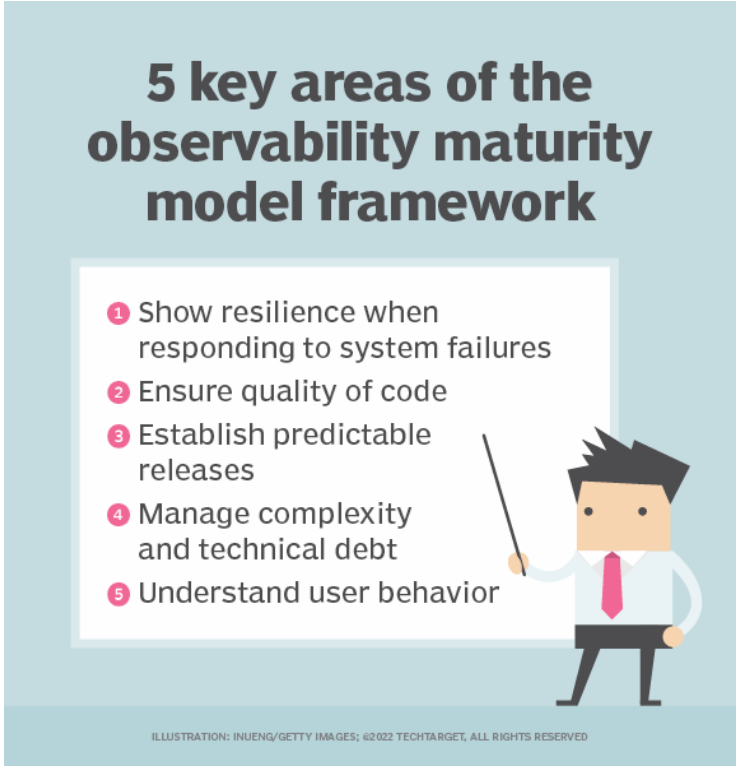
- **Accidental invisibility.** Failing to properly filter or structure data sources that compete for attention can lead to accidental invisibility of important events and data. This can cause a critical condition to be missed because it's hidden from view or processing.

- **Lack of source data.** Not all important information is collected, particularly at the application level and as it relates to tracing of workflows. Unlike resource or component status, traces of workflows usually require special software modifications to enable.
- **Multiple information formats.** It can be difficult to assemble the right information and interpret what's available when the same type of data comes in different formats from different sources. An organized strategy for structuring information into a standard form is required to ensure optimum observability handling.

HOW DO YOU IMPLEMENT OBSERVABILITY?

Observability starts with a plan, then moves to an architecture and finally to an observability platform. It's important to follow this approach or there's a greater risk of challenges and complications.

In this guide:



An observability plan begins by identifying the specific benefits desired. Then, it links each to a description of the data that would be needed to achieve it. While it's important that this linkage considers the available data from monitoring and telemetry, it's equally vital to identify important information that is not currently gathered -- or is gathered in a system that isn't contributing its data for observability analysis.

The observability architecture is a diagrammatic representation of the relationship between the source data and the presentation of data to operations personnel, AI and machine learning systems, etc. All data sources must be identified, along with the information that each source is expected to contribute. Above the data sources, the diagram should identify the tools that collect and present the information, the tool choices for data analysis and filtering, and the tool choices for data presentation. Both proprietary and open source tools for monitoring and observability are available; it's best to catalog the options that suit the specific target missions at this point.

The final step in implementation is a specific observability toolkit or an observability platform. The difference between the two can be subtle:

- A toolkit is a set of monitoring tools or features that can be used to support observability but rely on a human operator or a separate software layer to support collective analysis. A toolkit approach will usually require considerable customization but will accommodate existing software and data sources.
- An observability platform is an integrated software application that collects information, performs analysis that includes KPI derivations and presents actionable results to operations users. A platform might still require customization to accommodate all the data sources available, and it might also constrain the way data is integrated.

The value of observability depends on taking these three implementation steps in an organized way. Skipping or skimping will put the concept -- and the investment in it -- at risk.

CHOOSING OBSERVABILITY TOOLS AND DASHBOARDS

Observability is driven by data using logs, metrics and traces. Consequently, observability tools offer a wealth of features and functionality, each typically focusing on some aspect of IT and the applications environment. Regardless of the specific use case, typical observability tools share several common capabilities, and the tool should be able to perform these actions:

- generate native data, with or without the use of agents;
- ingest data produced from other monitoring or telemetry tools, such as logs;
- store and retrieve large volumes of data efficiently;
- process large volumes of disparate data to generate meaningful insights;
- visualize data and resulting analytics in configurable real-time dashboards; and
- produce meaningful reports, track long-term trends and send important alerts.

An observability tool can use these fundamental capabilities to offer a wide range of detailed services, such as the following:

- monitoring the infrastructure -- both locally and in a cloud -- to oversee the activities of containers, pods and networks, or aid in root cause analysis and fault isolation or troubleshooting;
- tracking applications and microservices to offer insight into application performance and availability or applications and infrastructure;
- supporting application security with vulnerability detection, alerting and even remediation; and
- offering business analytics and insights that correlate to the operational environment and projecting the risks and challenges of changes.

Organizations can choose from many available observability tools, but these are some of the most popular options:

- AppDynamics
- ContainIQ
- Datadog
- Dynatrace
- Grafana Labs
- Honeycomb
- Instana
- Lightstep
- LogicMonitor
- New Relic
- OpenTelemetry
- SigNoz
- Splunk

- Sumo Logic

Ultimately, the choice of observability tool depends on the organization's observability needs, integrations -- data sources -- and budget. Prudent IT and business leaders narrow the list by considering the specific feature set of each tool, and then test each final candidate in proof-of-principle projects before making a final tool implementation choice.

ENSURING OBSERVABILITY ACROSS YOUR ORGANIZATION

Making the decision to pursue observability is a good start, but actually ensuring observability can pose significant challenges for the business. Observability must ingest and sort through an enormous amount of data and then perform analytics to provide clear, actionable output. But the sheer volume of raw data -- especially from multiple sources -- makes analytics difficult, and resulting output has little value if it doesn't actually tell the business anything it wants to know. A business can boost the effectiveness and efficiency of their observability initiative through several practices, such as the following:

- **Set goals for observability.** Understand what's being observed and why, and what benefits are intended for the business through observability.
- **Curate data for observability.** Generate or ingest data that is relevant to the established goals, and don't bother with other nonessential data.

- **Optimize data for observability.** Review data sources and consider adding context or altering data collection to benefit observability, such as adding details to logs. This might include aggregating or rolling up some data to more easily see trends in a time series.
- **Seek meaningful and actionable outputs.** Details are easily lost in the noise of daily business, so look for meaningful data to produce actionable outputs, such as user effects to services and applications.
- **Configure outputs appropriately.** Configure reporting, alerting and dashboards to provide meaningful and actionable outputs. For example, rather than setting static alerting thresholds, configure time parameters that might forego an alert if the parameter returns to normal within a given time. This can cut down on unneeded noise.
- **Consider recipients.** Make sure that outputs are directed to proper channels. For example, reports might go to one admin, noncritical alerts might go to another admin and critical alerts might be directed to a third admin. This ensures the right people see the right outputs and nothing is ignored.

In this guide:

CONTINUED READING

Common use cases for observability

Observability vs. monitoring: What's the difference?

8 observability best practices