OpenTelemetry and the opportunity for intelligent observability

What's inside

INTRODUCTION

OpenTelemetry is emerging as the de facto instrumentation standard for cloud-native applications

As companies digitally transform and modernize applications, teams are increasingly adopting open source technologies to leverage new container- and microservice-centric architectures. This modular software approach improves agility, performance, and reliability. But it also creates new observability challenges, as companies often use different monitoring tools across clouds and applications.

OpenTelemetry is emerging as the de facto standard for adding observable instrumentation to cloud-native applications.

The open source framework provides a common mechanism for collecting critical application and infrastructure telemetry data, such as metrics, traces, and logs. Enterprises implementing OpenTelemetry can use that data to gain complete, AI-driven observability at scale through integration with Dynatrace.

An overview of OpenTelemetry

What it is
OpenTelemetry is an open source observability framework for cloud-native software that was created from the merger of the OpenTracing and OpenCensus projects. A Cloud Native Computing Foundation incubating project, OpenTelemetry is the second most popular project behind Kubernetes.

What it does
OpenTelemetry provides a standard way to instrument, generate, collect, and export telemetry data, including metrics, logs, and traces. While it is a tool for capturing, transmitting, and parsing telemetry data, it does not provide back-end storage or analytics.

A typical use case involves instrumenting applications with OpenTelemetry and then sending the resulting telemetry to a third-party observability solution for further analysis and visualization.

Where it's headed
With an initial 1.0 release of tracing application programming interfaces (APIs) and software development kits (SDKs) in February 2021, OpenTelemetry is under active development with rolling release cycles. Of the three telemetry capabilities, traces are the most production-ready, with metrics GA expected in mid-2022. As a result, most OpenTelemetry implementations today focus on application tracing.

While nascent, industry adoption of OpenTelemetry is expanding.

Amazon Web Services
- Announced support
- Launched a distribution for OpenTelemetry

Microsoft Azure
- Announced support
- Released an OpenTelemetry exporter for Azure Monitor

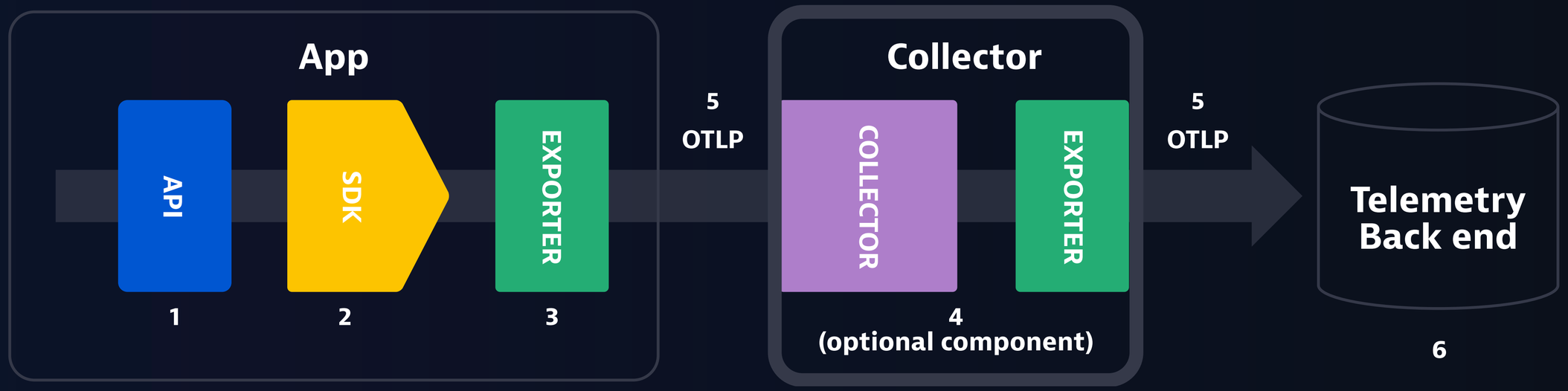
Google Cloud
- Announced support
- Released an OpenTelemetry exporter for Google Cloud Trace

Software-as-a-service (SaaS) vendors are also starting to explore adding OpenTelemetry to their platforms to allow customers to access SaaS telemetry data.

SECTION 1

OpenTelemetry architecture

OpenTelemetry consists of several components, as depicted below. Here's a high-level look at each one, from left to right:



1.

APIs are a core component of OpenTelemetry. These are language-specific — Java, Python and .NET, for example — and provide the basic "plumbing" for instrumenting the code to generate telemetry data, such as metrics, logs, and traces.

2.

The **SDKs** are another language-specific component. The implementation of the OpenTelemetry API serves as the bridge between the APIs and the exporter. This component allows for additional configuration, such as request filtering and transaction sampling.

3.

The **in-process exporter** is part of the SDK.

- Allows configuration to which back end(s) the telemetry data is sent
- Decouples the instrumentation from the back-end configuration
- Makes it easy to switch back ends without the pain of re-instrumenting your code

4.

The **collector**, while not technically required, is a useful component of the OpenTelemetry architecture.

- Allows greater flexibility for receiving, transforming, and sending the application telemetry to the back end(s)
- Is a standalone process that serves as a centralized location, receiving, processing, and exporting telemetry data
- Becomes especially important in enterprise environments where there can be many firewalls

The collector has two deployment models:

The first is an **agent** that resides on the same host as the application reporting data to it — binary, daemonset, or sidecar, for example. This collector can then send data to a server, operating system, database, or network directly or via another collector.

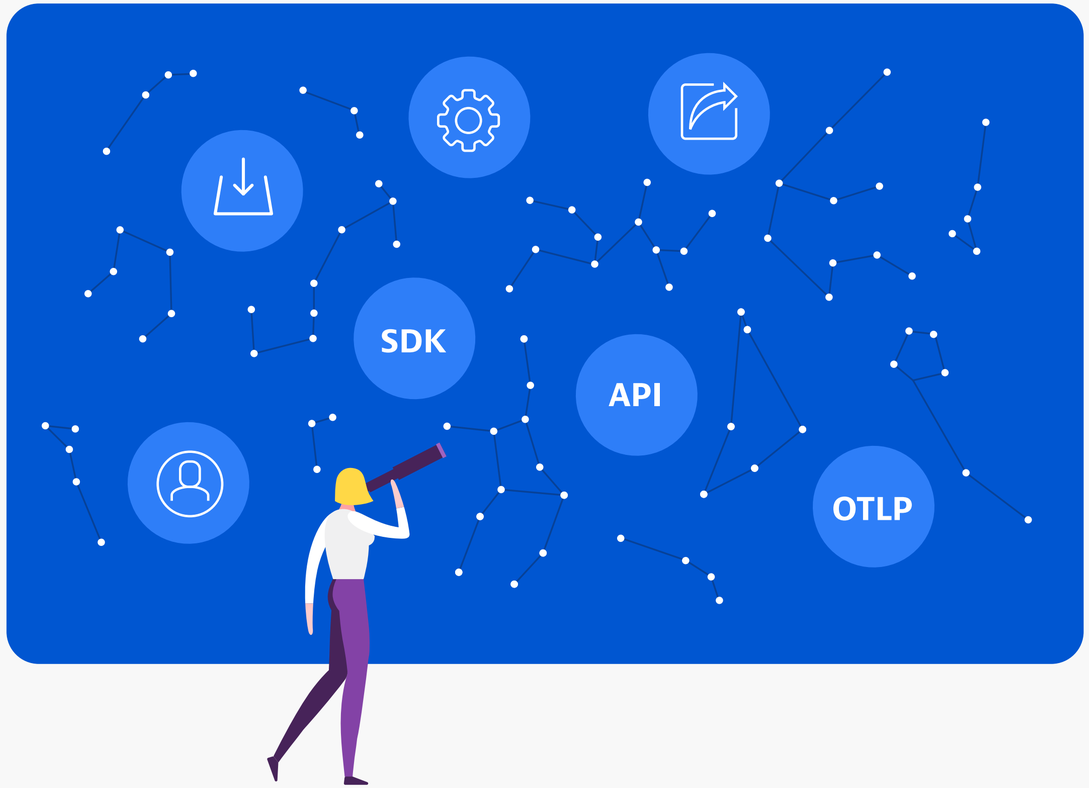
The second is a standalone process (**gateway**) completely separate from the application(s) reporting telemetry data to it. It's responsible for exporting this data to a back-end observability tool.

5.

OpenTelemetry Protocol (OTLP) is the OpenTelemetry native format that supports metrics, logs, and traces in a single data stream. Exporters and collectors can be used to translate OTLP into the language of the back-end destination and then transport the data there.

6.

Because OpenTelemetry is just a specification about collecting and sending telemetry, it still requires a **telemetry back end** to receive and store the data. There are numerous back ends available with different capabilities depending on your specific needs.



SECTION 2

The benefits of OpenTelemetry

Standard solution API

Before:
Prior to OpenTelemetry, the process of capturing and transmitting application telemetry data was fragmented with a range of disparate open and vendor-provided solutions.

With OpenTelemetry:
OpenTelemetry standardizes the telemetry API for metrics, logs, and traces. This open, standards-based approach enables broader adoption across a range of application development and operations teams. By providing a common approach to telemetry, it also enables collaboration between development and operations teams.

Vendor agnostic

Before:
Traditional approaches to telemetry data collection required teams to manually instrument code, which locked users into specific tools. As a result, the cost of switching was high. So, if a customer adopted a solution that worked for monolithic apps but not cloud apps, moving to a more modern offering was challenging.

With OpenTelemetry:
OpenTelemetry separates telemetry data collection from analytics. Because data is transmitted using OpenTelemetry's standards, customers can choose any back-end analytics platform, assuming it supports OpenTelemetry. This is particularly important as modern microservices-based applications have very different observability requirements than traditional monolithic architectures.

Pre-instrumented code

Before:
Application tracing is critical to help users identify application issues and areas for improvement. Manually instrumenting code to export trace data can be challenging because it can require extensive testing cycles and code validation. With SaaS software, manual instrumentation may not even be possible.

With OpenTelemetry:
OpenTelemetry provides a standard method for developers to expose tracing information from their software libraries and SaaS applications. As a result, users adopting these pre-instrumented libraries and applications can access OpenTelemetry telemetry data, which users can store, manage, and analyze in their observability platform.

Open source

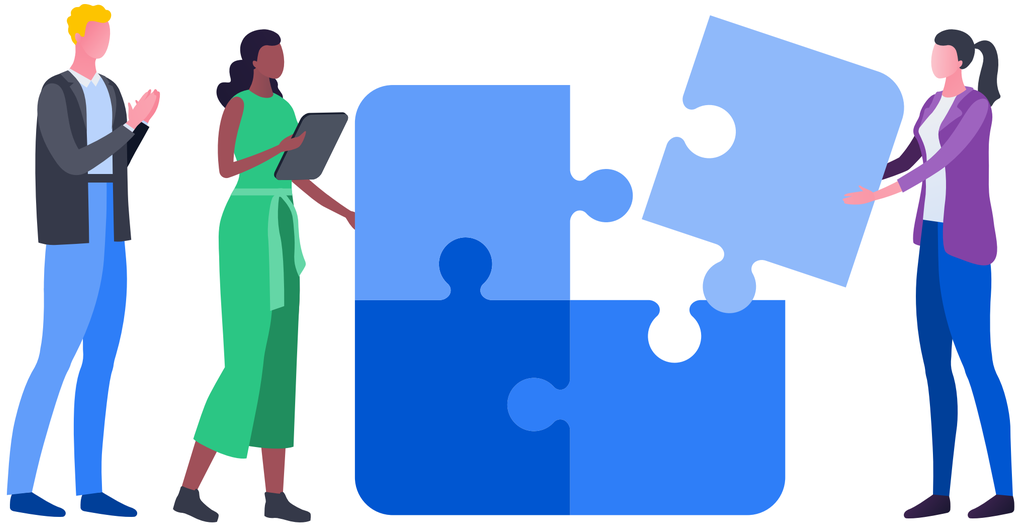
The open source development model provides several advantages, including the following:

- It's vendor-agnostic
- It encourages practitioner, vendor, and developer collaboration
- It maintains transparency with publicly available source code

Community collaboration is particularly useful for OpenTelemetry. By working together, the community can create a standard approach to pre-instrumentation available to everyone, which saves time and money.

Standard methodology for adding business-specific data

OpenTelemetry enables users to add information to telemetry data using key/value pairs, known as attributes. With this method, OpenTelemetry provides a vendor-agnostic way to add business-specific information to individual tracing spans, like a user's status level, an A/B testing marker, or how much money was processed in a transaction.



SECTION 3

OpenTelemetry considerations

While OpenTelemetry has no shortage of benefits, there are factors to consider before bringing the framework to your enterprise.

Security

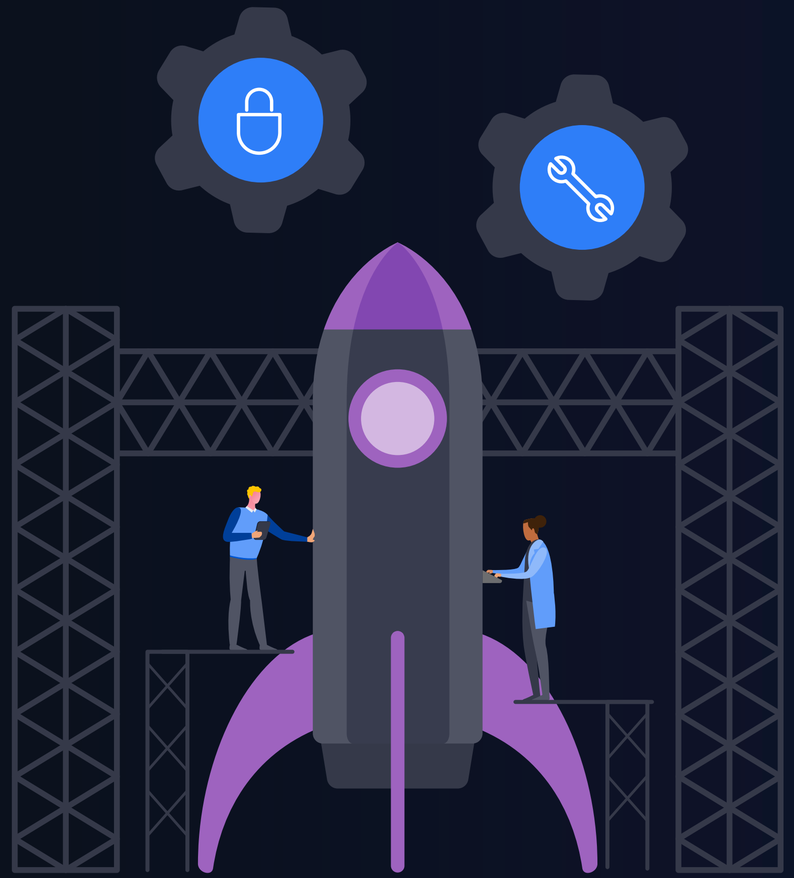
Application security is more important than ever, and an unexpected code or library vulnerability can have disastrous consequences.

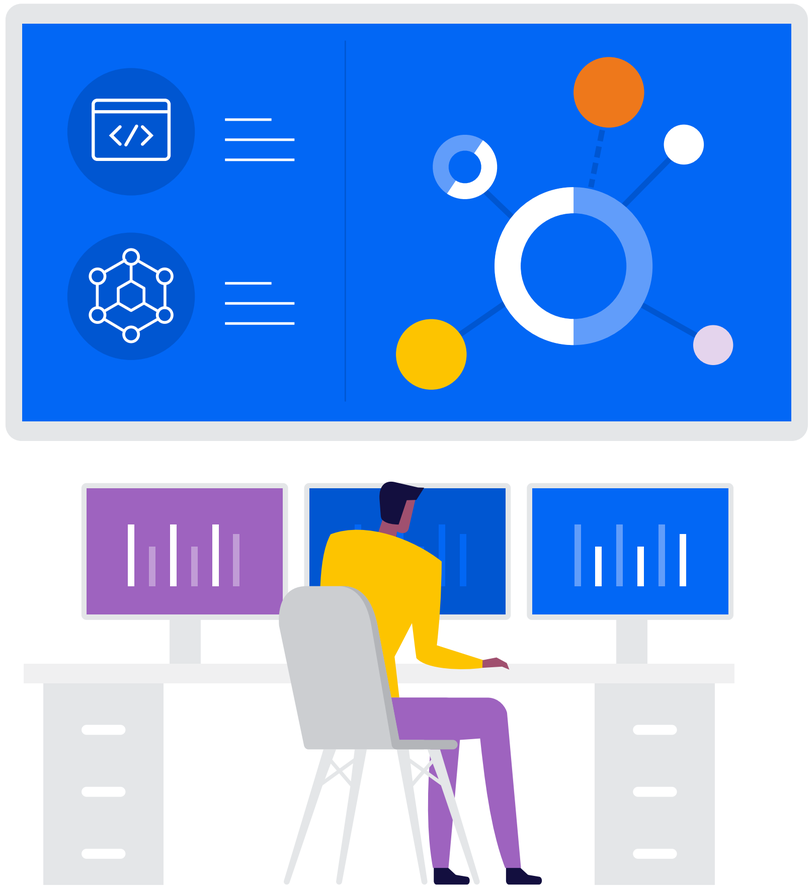
As an open source project, OpenTelemetry is evolving rapidly and accepting contributions from hundreds of volunteer developers. Security is a critical component of the project, but maintaining enterprise-level security can be challenging with a diverse range of contributors. Some third-party OpenTelemetry distributions may provide added security validation, but commercial vendors typically invest more time and resources into these endeavors.

Supportability

In complex modern application environments, there's always potential for an unexpected issue that affects application performance or availability. It's essential to have access to product experts who can accelerate issue resolution.

Open source technologies such as OpenTelemetry rely on a distributed development model and encourage contributions from a range of practitioners, vendors, and developers. However, this same flexibility introduces support challenges. You can file an issue on GitHub, but there are no support service-level agreements in place. Therefore, problem resolution is typically on a best-effort basis.





Breadth of supported languages and protocols

Companies are in different stages of modernization and typically operate a heterogeneous environment incorporating hundreds of applications, microservices, and programming languages running on premises and in the cloud. Breadth of support is critical to ensure a unified monitoring and tracing experience.

OpenTelemetry started with only a handful of language-specific implementations as production-ready for tracing (Stable or 1.0), including C++, .NET, Java, JavaScript, and Python. This list is continually growing. There are alpha, beta, and release candidate implementations for more than a dozen languages, with support in progress for metrics, logging, and tracing.

Before implementing, you need to consider which languages and versions to use and how they align with OpenTelemetry's support matrix. Additionally, you should consider the status of metrics and log support, as those capabilities are still in development.

Version management

A strong benefit of open source software is its continuous and rapid development. However, this change rate creates challenges, as new releases can add or deprecate features that undermine production environments. In an ideal world, a company deploys one version of OpenTelemetry to ensure customers use common experiences and learnings across all implementations.

Because OpenTelemetry is freely downloadable, it can be challenging to maintain strict version controls, as developers and users can simply download and install the latest release. Additionally, developers may want to implement detailed qualification processes to validate each new OpenTelemetry instance.

Instrumentation

There are multiple ways to instrument code. The two most common approaches are the following:

Auto-instrumentation. An approach in which instrumentation is applied with no code changes using an agent

Manual instrumentation. An approach in which instrumentation requires code modification, such as source-code changes or adding new, pre-instrumented libraries

OpenTelemetry is embarking on a journey where instrumentation will not require code modification. Although initially, only a few languages are implemented using an auto-instrumentation approach, including Java and .NET.

For many organizations, implementing manual instrumentation can consume significant amounts of development and QA time. All applications must be modified, which results in qualification and validation cycles and creates a slow mean time to observability. This is a common reason customers prefer a no-code auto-instrumentation strategy.

Infrastructure

OpenTelemetry deployments typically require an additional component, called a collector. The number of collectors installed depends on the size of the environment and the amount of data. In practice, most environments will have at least one. An end user must continually optimize collector usage — for example, determine how many collectors are needed — and monitor, manage, and update each instance.

While monitoring an individual collector can be easy, as environments expand, monitoring a growing fleet of instances can add to cost and complexity.

Observability limitations

Modern applications can generate vast volumes of telemetry data. While OpenTelemetry provides a standard approach that simplifies data capture, it does not help users better understand how to optimize application performance and availability based on that data.

Users must purchase and manage other solutions to store and analyze the generated data.

SECTION 4

Automatic and intelligent observability with Dynatrace and OpenTelemetry

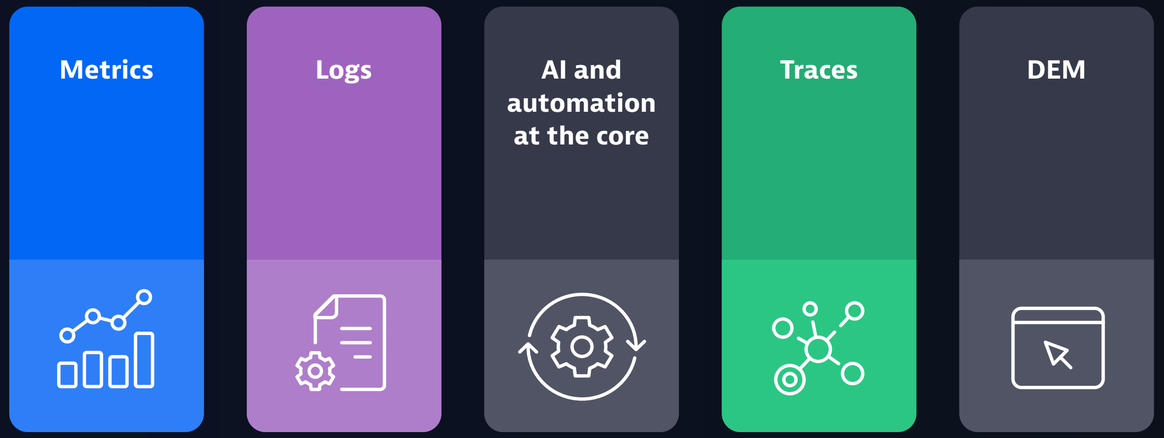
OpenTelemetry certainly has its benefits, but it's not an observability platform. While capturing telemetry data using a common framework is beneficial, you still have to use that data to get insights and answers. Dynatrace's AI-driven Software Intelligence Platform makes the most of OpenTelemetry with features like AI-powered root-cause analysis, real-user monitoring, synthetic transaction monitoring, auto-discovery, topology mapping, and more.

Three pillars of observability: The missing pieces

Metrics, traces, and logs are the three pillars, but they're just data. In modern dynamic environments, simply having the data is not enough. You also need the following:

- Automation to ensure you're capturing data from the most short-lived services
- AI to help sift through the data to get answers
- Digital experience management to help you understand how users are affected

You can only achieve complete observability through all of these pillars. Observability powered by AI and automation allows you to spend less time on repetitive manual tasks, war rooms, and finger-pointing, leaving more time for innovation.



Enterprise-class platform

While OpenTelemetry provides a standard way to capture, transmit, and parse telemetry data, it still requires a back-end system to store and visualize application data. The dynamic nature of modern cloud environments makes it critical to combine OpenTelemetry with an observability platform that delivers enterprise capabilities, such as:

Security

While OpenTelemetry was built with security in mind, it's only as secure as the platform with which it's paired. The Dynatrace platform was built from the ground up with a focus on enterprise security. It includes numerous security-centric features such as single sign-on, flexible deployment options, detailed audit logs, and data segregation. The platform also incorporates multiple certifications, including FedRAMP, SOC 2, Type II, independent penetration testing, and CSA certification.

With the growing concern about vulnerabilities in open source software, Dynatrace Application Security provides automatic detection and mitigation for vulnerabilities detected in production and pre-production environments.

Privacy

Data privacy is a critical concern, particularly with compliance frameworks like the General Data Protection Regulation (GDPR). Violations can result in significant financial penalties and reputational damage. The Dynatrace Software Intelligence Platform incorporates a broad range of privacy capabilities that enable customers to maintain GDPR and other regulatory compliance while still benefiting from the platform's deep insights.

Critical capabilities include:

- Masking confidential personal data
- Access controls to limit specific data access to administrators with appropriate privileges
- Configurable data retention policies
- Much more

These capabilities complement OpenTelemetry and enable companies to meet more stringent regulatory requirements while reaping the benefits of open source.

Scale

Given the increasing volume, variety, and velocity of telemetry data created in cloud-native environments, you need a technology like Dynatrace's Software Intelligence Platform to grow with your data and bring AI- and automation-driven real-time answers. This is particularly critical with modern, microservices-based application architectures, which are dynamic, ephemeral, and generate larger quantities of telemetry data.

Automation

With dynamic, cloud-native environments, services and functions constantly spin up and down. Traditional approaches that use manual discovery and instrumentation struggle to keep up, introducing complexity and cost. You need a different approach.

The Dynatrace Software Intelligence Platform can automatically discover and capture telemetry data from all cloud and traditional services and from multiple data sources, including OpenTelemetry.

Intelligence

Modern microservices-based cloud applications generate massive amounts of telemetry data between metrics, logs, and traces. While OpenTelemetry standardizes the collection and transmission of this information, capturing the data is not enough. You need to identify potential issues and the appropriate actions to optimize application performance and availability.

The sheer quantity of data is beyond what a human can manage, and Dynatrace's deterministic AI is mandatory to identify issues and required corrective actions. Dynatrace AI offers precision you can rely on, including the following:

- Dependency detection
- Topology visualization
- Anomaly detection and prioritization
- Context-based root-cause analysis with business impact analysis
- AIOps to automate operations and enable self-healing

Flexible deployment options

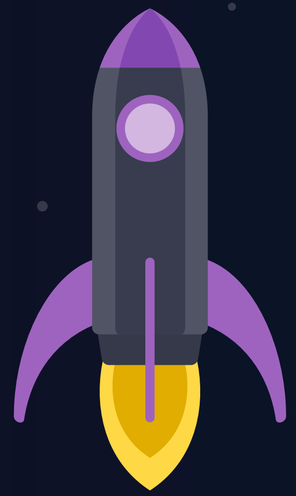
OpenTelemetry is a young project under rapid development. While it has an extensive roadmap, the maturity and breadth of application support can vary. Additionally, the requirement to modify application code to deploy the technology can be a hindrance.

Dynatrace OneAgent captures observability data without any code changes and introduces additional management capabilities, like version control and automatic updates.

Customers can choose Dynatrace, OpenTelemetry, or both to capture their observability telemetry data with the confidence that the full breadth of Dynatrace's AI and automation capabilities apply to all data types. It also enables a customer's journey to OpenTelemetry, as they can rely on Dynatrace auto-instrumentation while going through the process of reworking their applications.

Front-end monitoring

As an observability framework focused on metrics, logs, and traces, OpenTelemetry provides the key pillars of observability. However, to prioritize and make sense of the data, you need to understand the user context. This includes user experience data to understand how application changes affect customer outcomes. Dynatrace fully supports OpenTelemetry and extends it to include end-user monitoring, providing a complete, front-to-back view.



Commitment to OpenTelemetry

Dynatrace not only supports OpenTelemetry in our product, but we are also contributing our 20 years of observability experience to the project. Dynatrace is a top committer to OpenTelemetry and will continue to partner with the open source community to deliver further enhancements.

Learn more

Dynatrace (NYSE: DT) exists to make the world's software work perfectly. Our unified software intelligence platform combines broad and deep observability and continuous runtime application security with the most advanced AIOps to provide answers and intelligent automation from data at enormous scale. This enables innovators to modernize and automate cloud operations, deliver software faster and more securely, and ensure flawless digital experiences. That's why the world's largest organizations trust Dynatrace® to accelerate digital transformation.

Curious to see how you can simplify your cloud and maximize the impact of your digital teams? Let us show you. Sign up for a free 15-day Dynatrace trial.

dynatrace

06.24.22 16927_EBK_agency/jw/cs