OpenTelemetry

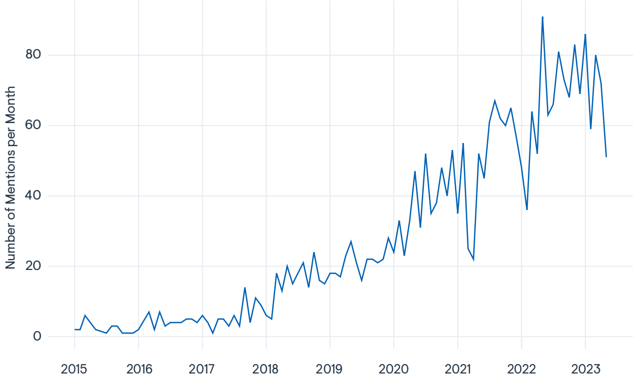
Prepared for Chronosphere
By **Torsten Volk**, Managing Research Director, *Hybrid Cloud, Software-Defined Infrastructure, and Machine Learning*

EMA
IT & DATA MANAGEMENT RESEARCH,
INDUSTRY ANALYSIS & CONSULTING

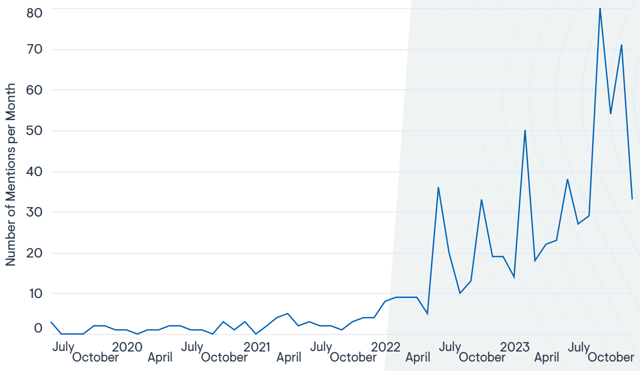
The burgeoning interest in observability, as evidenced by the sheer number of related repositories on GitHub, underscores the industry's collective push toward more transparent and manageable systems. OpenTelemetry, with its comprehensive set of APIs, libraries, agents, and instrumentation, is strategically positioned to cater to this demand.

The emphasis on Kubernetes observability, in particular, plays into OpenTelemetry's strengths. As organizations increasingly adopt Kubernetes for container orchestration, the need for a unified and standardized observability solution becomes paramount. OpenTelemetry's compatibility with Kubernetes and its integration with popular tools like Prometheus offer a seamless observability experience for developers and operators alike. Moreover, the project's commitment to providing a single set of APIs and instrumentation for both tracing and metrics ensures that users don't have to juggle multiple tools or face integration challenges. This holistic approach to observability, combined with the project's open source nature, encourages widespread adoption and community contributions.

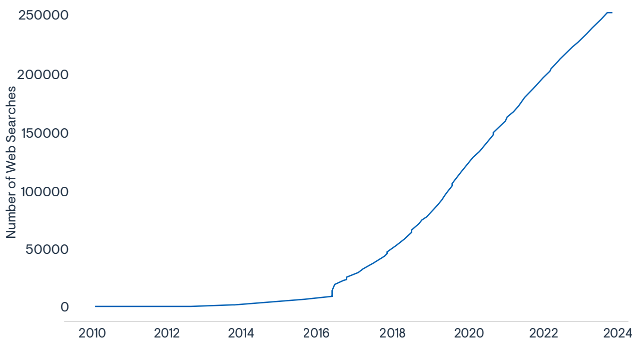
OBSERVABILITY-RELATED BUSINESS DISCUSSIONS ON Y-COMBINATOR



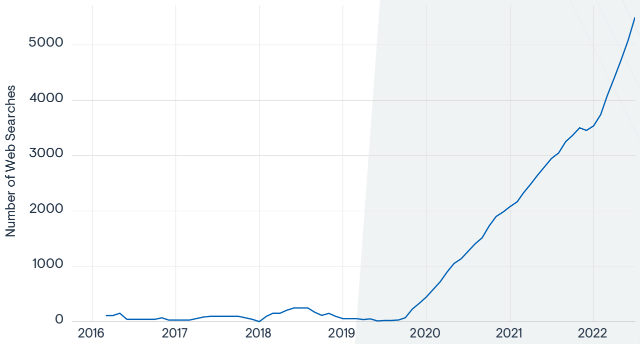
OPENTELEMETRY-RELATED BUSINESS DISCUSSIONS ON Y-COMBINATOR



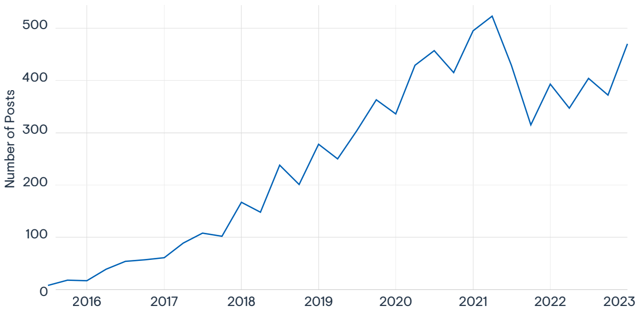
MONTHLY OBSERVABILITY-RELATED WEB SEARCH TERMS



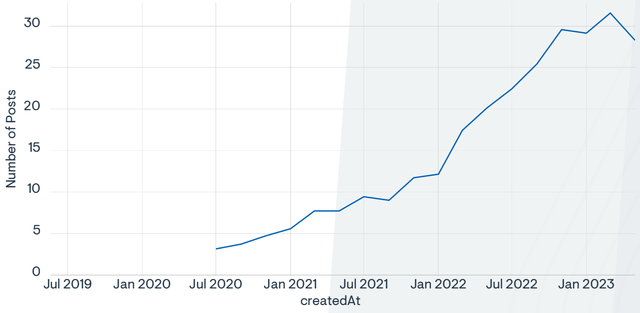
MONTHLY OPENTELEMETRY-RELATED WEB SEARCH TERMS



MONTHLY OBSERVABILITY-RELATED DISCUSSIONS ON KUBERNETES SLACK



MONTHLY OPENTELEMETRY-RELATED DISCUSSIONS ON KUBERNETES SLACK

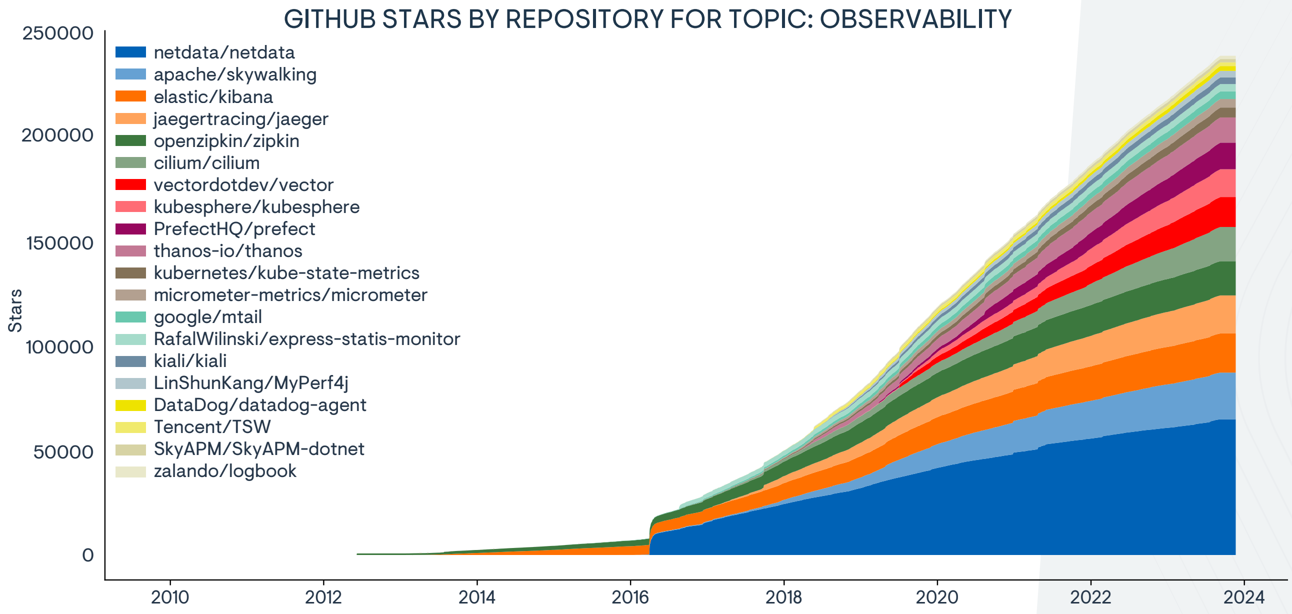


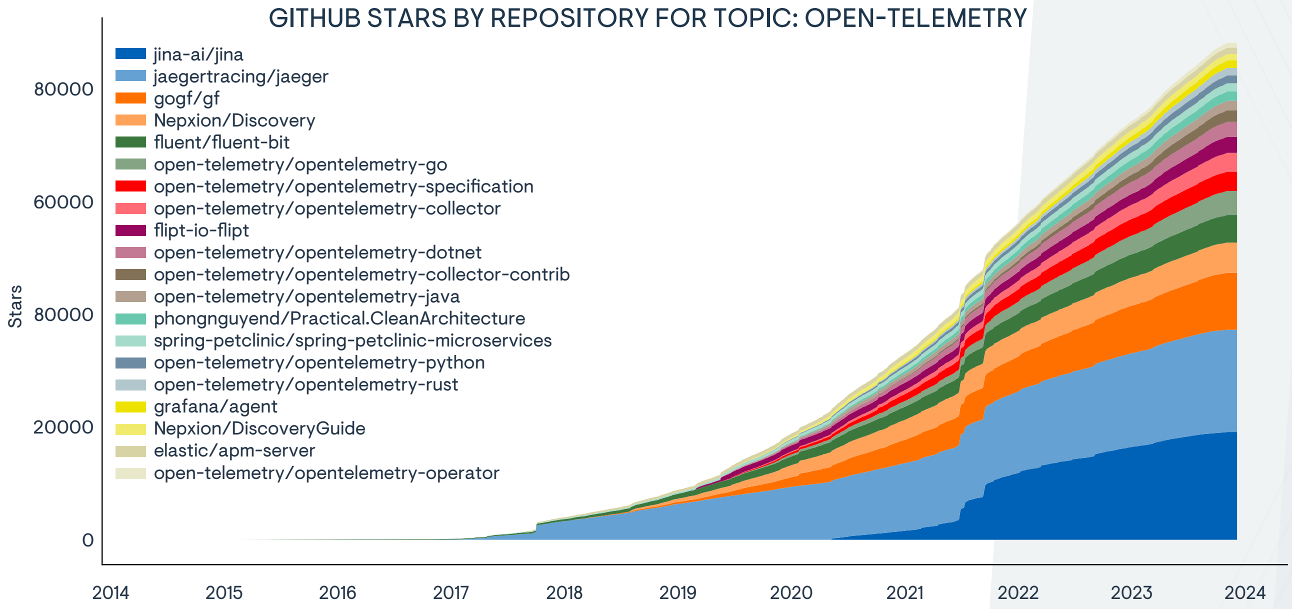
Source: Serverfault

Furthermore, the convergence toward standard practices in the observability space, as indicated by the popularity of topics like “metrics” and “OpenTelemetry” among the top GitHub repositories, provides a tailwind for OpenTelemetry’s growth. As the industry seeks standardized solutions to avoid fragmentation and ensure interoperability, OpenTelemetry’s vision of a unified observability platform resonates strongly. In essence, the current trajectory of the observability domain, with its focus on real-time monitoring, Kubernetes, and standardization, amplifies the relevance and success of OpenTelemetry in the modern software ecosystem.

OpenTelemetry: How it Works

OpenTelemetry provides a standard set of APIs that define the basic functionalities for tracing, metrics, and baggage. “Baggage” refers to the key value pairs that can be added to spans and are propagated in-band along with the trace context. This allows you to annotate your traces with custom data that can be used for analysis later. The APIs are implemented by SDKs, which are language-specific libraries that offer a range of configuration options.


GITHUB STARS BY REPOSITORY FOR TOPIC: OBSERVABILITY


GITHUB STARS BY REPOSITORY FOR TOPIC: OPEN-TELEMETRY

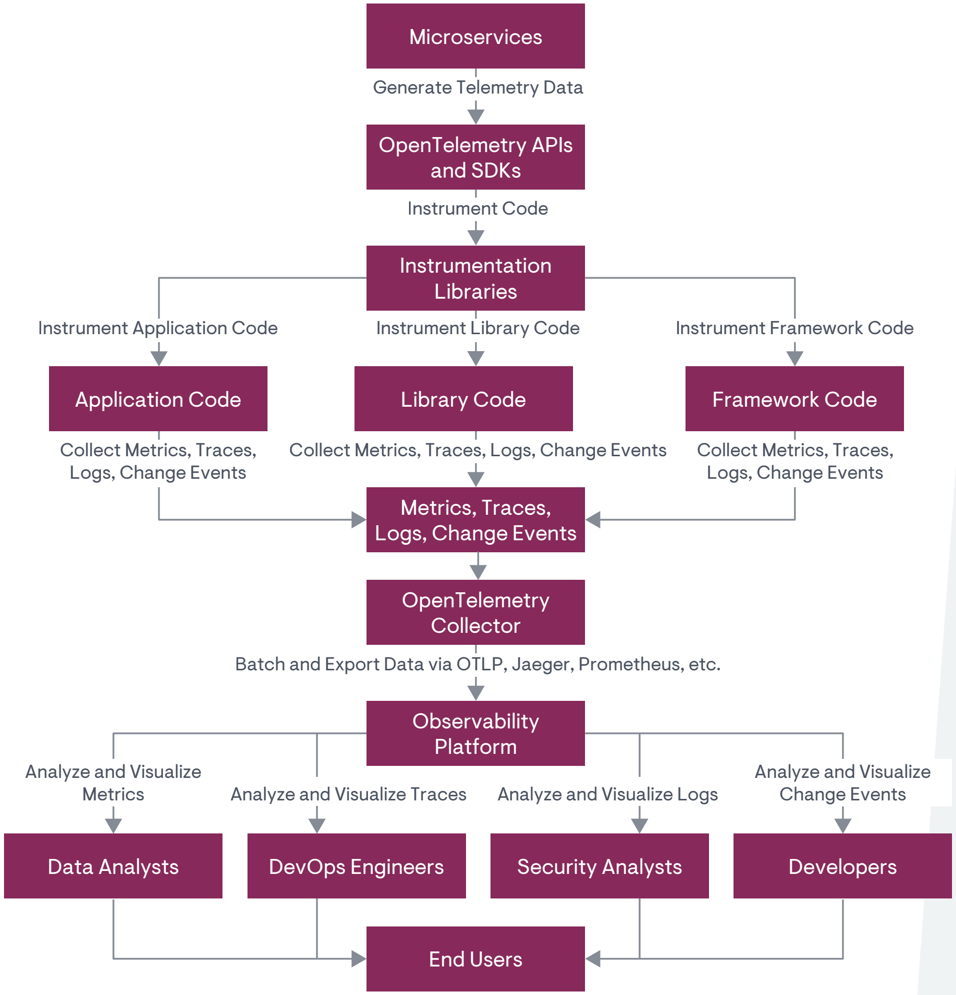
SDKs serve as the foundation for instrumentation libraries, which automatically collect data from popular frameworks and libraries. They also configure samplers, which decide which traces to collect based on predefined rules. SDKs also hook into span processors, which allow you to add custom logic or attributes to spans as they are started and ended. This is useful for adding business-specific information to your traces.

Observers are another feature of SDKs; they are used for reporting asynchronous metrics. These are metrics that are not initiated by the application but are pulled from the system, like CPU or memory usage. Context propagation is also enabled by SDKs; this is the mechanism that allows trace context to be passed between services in a distributed system, ensuring that all the spans from the same trace are connected.

The collector is a pivotal component that receives data from instrumentation libraries and uses context propagation to correlate events across multiple services. It employs processors for tasks like data filtering and aggregation and extensions for additional functionalities, such as health checks. The collector then exports the data through exporters, which handle a variety of tasks from data transformation and serialization to transmission to backend systems.

Additional components, like resource detectors, enrich the data with metadata. Integrations facilitate compatibility with other observability tools. Client libraries are used for collecting data from frontend or mobile applications. All these components are supported by extensive documentation and can contribute to or utilize the contrib repository, which is a community-maintained collection of extensions and integrations.

In summary, the flowchart provides a holistic view of OpenTelemetry's complex architecture, detailing how data flows from collection to export. It also highlights the framework's extensibility and the customization options available at different stages.


Microservices
Generate Telemetry Data
OpenTelemetry APIs and SDKs
Instrument Code
Instrumentation Libraries
Instrument Application Code
Instrument Library Code
Instrument Framework Code
Application Code
Library Code
Framework Code
Collect Metrics, Traces, Logs, Change Events
Collect Metrics, Traces, Logs, Change Events
Collect Metrics, Traces, Logs, Change Events
Metrics, Traces, Logs, Change Events
OpenTelemetry Collector
Batch and Export Data via OTLP, Jaeger, Prometheus, etc.
Observability Platform
Analyze and Visualize Metrics
Analyze and Visualize Traces
Analyze and Visualize Logs
Analyze and Visualize Change Events
Data Analysts
DevOps Engineers
Security Analysts
Developers
End Users