<https://www.splunk.com/en_us/blog/learn/observability.html#:~:text=Simply%20put%3A%20Observability%20is%20the%20ability%20to%20measure,only%20using%20information%20from%20outputs%2C%20namely%20sensor%20data>.

**What is Observability? Introduction!**

Observability in containers and microservices exposes the state of

applications in production.

By this developers can better identify and resolve performance issues.

Container services (such as Docker, Kubernetes and others) and

microservices address the increased risk of downtime and other

issues related to cloud environments or monolithic software,

in which any change to the single codebase affects the entire

application and its dependencies.

Containers and microservices break applications down into

independent services, allowing developers to modify and

redeploy a particular service rather than the whole application.

However, a container-based architecture introduces new challenges.

Interdependent microservices are typically scattered across

multiple hosts, and as the infrastructure scales, so does the

number of microservices in production.

This makes it difficult for DevOps teams to know what’s currently

running in production, leading to longer delivery cycles,

downtime and other issues.

Observability addresses these challenges,

providing visibility into distributed systems that help developers

better understand an app’s performance and availability.

In the event of a failure, it provides the control needed to

pinpoint bottlenecks and debug or fix the problem quickly.

-----------------------------------------------------------------

**How to implement Observability?**

Observability uses three types of telemetry data — metrics, logs and

traces — to provide deep visibility into distributed systems

and allow teams to get to the root cause of a multitude of issues

and improve the system’s performance.

Primary Data Classes Used in Observability

The primary data classes used in observability are

logs, metrics and traces.

Together they are often called “the three pillars of observability.”

Logs: A log is a text record of an event that happened at a

particular time and includes a timestamp that tells when it

occurred and a payload that provides context.

Logs come in three formats:

Plain text

Structured

Binary

Plain text is the most common, but structured logs — which include

additional data and metadata and are easier to query — are becoming

increasingly popular.

Logs are also typically the first place you look when something goes

wrong in a system.

Metrics: A metric is a numeric value measured over an interval of time

and includes specific attributes such as timestamp, name, KPIs and value.

Unlike logs, metrics are structured by default, which makes it easier

to query and optimize for storage, giving you the ability to retain

them for longer periods.

Traces: A trace represents the end-to-end journey of a request

through a distributed system. As a request moves through the host

system, every operation performed on it — called a “span” — is encoded

with important data relating to the microservice performing that operation.

By viewing traces, each of which includes one or more spans,

you can track its course through a distributed system and identify

the cause of a bottleneck or breakdown.

Integrating the three pillars: Working with these data classes doesn’t

guarantee observability, particularly if you’re working with them

independently of each other or are using different tools for each function.

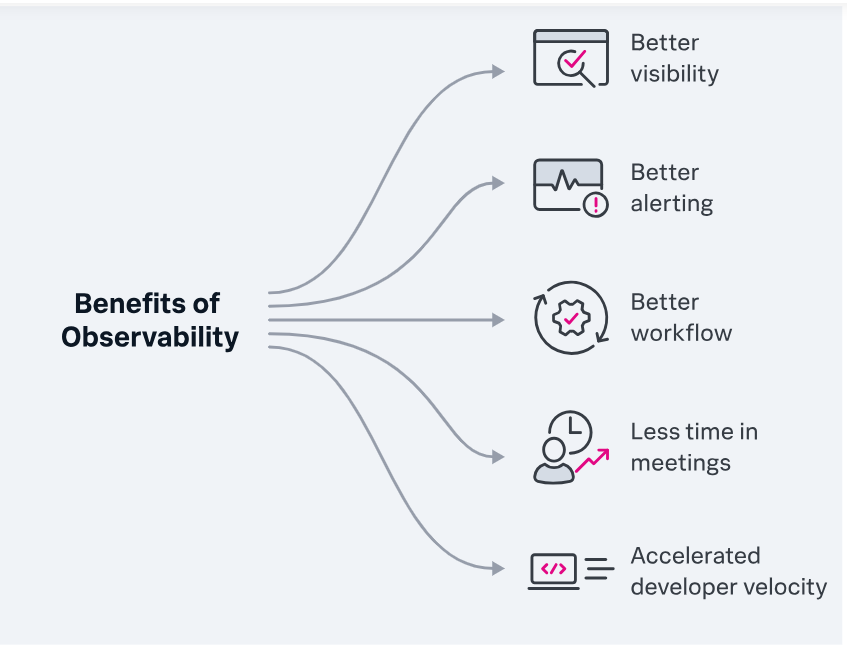
Rather, you’ll achieve a successful approach to observability by

integrating your logs, metrics and traces within a single solution.

When you do this, you not only understand when problems occur,

but understanding why those problems are occurring.

**Benefits of Observability (to the DevOps team):-**



**Tools that implement Observability should provide for the following:**

To achieve observability you need proper tooling of your systems and apps to collect the appropriate telemetry data. You can make an observable system by building your own tools, using open source software or buying a commercial observability solution. Typically there are four components involved in implementing observability:

* Instrumentation: These are measuring tools that collect telemetry data from a container, service, application, host and any other component of your system, enabling visibility across your entire infrastructure.
* Data correlation: The telemetry data collected from across your system is processed and correlated, which creates context and enables automated or custom data curation for time series visualizations.
* Incident response: [Incident management](https://www.splunk.com/en_us/blog/learn/incident-management.html) and automation technologies are intended to get data about outages to the right people and teams based on on-call schedules and technical skills.
* [AIOps](https://www.splunk.com/en_us/blog/learn/aiops.html): Machine learning models are used to automatically aggregate, correlate and prioritize incident data, allowing you to filter out alert noise, detect issues that can impact the system and accelerate incident response when they do.

