Now this doesn't mean that you always need an incident to review dashboards. You can also explore the current health of the system or the dullest applications seem exciting with their graphs and colors, you shouldn't be sitting around watching dashboards and waiting for... 

So this is where we finally get to use our observability, identify issues, notify the software team of those issues, and even predict potential trends in our system's reliability. 

Now that we have a good understanding of observability, what about monitoring? With monitoring, we use some of our observability tools to monitor, no matter how many fancy dashboards you may have. 

Beware the Three Pillars

Beware the Three Pillars

Logs

Next, let’s look at logs. Logs provide textual data regarding events that occur in your system. They are the most direct way to view occurrences within your application. Logs are critical for troubleshooting because they often contain error messages and other information that can help identify the root cause of a problem. 

For our third pillar, let’s talk about traces. 

Traces

Traces are an essential aspect of observability. They provide insight about how to get the most out of your logs.

With logs, consider using best practices such as:

- **Log Level**: Use different log levels (e.g., debug, info, warning, error) to categorize log messages based on their severity.
- **Log Rotation**: Regularly rotate logs to avoid filling up disk space and to keep track of log data over time.
- **Log Retention**: Set retention policies to keep logs for a specific period and then delete them to manage storage.
- **Log Aggregation**: Use log aggregation tools to consolidate logs from different sources into a single view.

Gauge

Gauge is a type of metric that represents a single value at a point in time. It is used to monitor a system's health, such as a count of incidents or a sum of transactions. 

Histogram

A histogram is a visualization technique used to represent the distribution of a set of numerical data. It is particularly useful for understanding the frequency of occurrence of certain values or ranges within a dataset. 

In control theory, you need both monitoring and observability if you want to build reliable systems. Observability helps teams resolve the problems by improving debugging and cause analysis. Additionally, monitoring uses observability to detect issues and notify the software team of those issues. 

In this time? That’s helpful info, but it doesn’t provide the ability to trace one particular transaction or customer until we add tracing. 

Both logs and metrics can relate to particular events that occur within the system. What happened at this particular place in our system at this time? That’s helpful info, but it doesn’t provide the ability to trace one particular transaction or customer until we add tracing. 

We’re going to go a bit further with metrics, as they vary in type. The following are the most common, though you may see others.

**Counter**

Counters measure events that occur. For example, you may count the number of requests your API receives, the number of purchases made on your e-commerce platform, or the number of page views on your website. 

**Gauge**

Gauges measure a single value at a point in time. They are often used to monitor system health, such as the number of active users, the amount of memory used, or the number of requests in the queue. 

**Histogram**

As a combination of the gauge and histogram, here you can see the distribution of gauge data. So if we take queue experience.

And instead of using just an average or max, you can determine the 50th, 90th, or 99th percentile of latency that your customers will experience.

So that’s it. That’s the simple answer. But as you may expect, more goes into it than that. Let’s look further.

You need both monitoring and observability if you want to build reliable systems. Observability helps teams resolve the problems by improving debugging and cause analysis. Additionally, monitoring uses observability to detect issues and notify the software team of those issues. 

In control theory, you need both monitoring and observability if you want to build reliable systems. Observability helps teams resolve the problems by improving debugging and cause analysis. Additionally, monitoring uses observability to detect issues and notify the software team of those issues. 

Let’s start with a simple explanation.

**Observability vs. Monitoring**

When someone mentions observability and monitoring, we often picture rows of dashboards and radiators providing critical data to a team of analysts. This is certainly one application of monitoring, but it's only one of many.

**Monitoring**

Monitoring is the process of gathering and interpreting data to detect issues in the system. It involves observing the current state of the system and alerting on any deviations from expected behavior. 

To put it simply, monitoring helps teams identify problems and receive notifications about them. To follow problem identification, you need both monitoring and observability if you want to build reliable systems. Observability helps teams resolve the problems by improving debugging and cause analysis. Additionally, monitoring uses observability to detect issues and notify the software team of those issues. 

**Observability**

Observability is the capability to understand the behavior of your system. It involves gathering and analyzing data to understand the system's current state, past behavior, and future trends. 

Observability helps teams understand what's happening within the system, allowing them to identify issues, notify the software team of those issues, and even predict potential trends in the system's reliability.

So now we have a high-level view of the three pillars. But you should use some caution when determining whether you have observability.

Beware the Three Pillars

Beware the Three Pillars