Observability - A complete Guide: Part 1

What is Observability?

Observability refers to the internal state of your system based on the data it produces. Here the system refers to the application, the infrastructure used for setting up this application, and the network. All these components are part of the system. So if observability is in place then you can understand the state of the application, the state of the infrastructure and understand the state of the network.

But observability goes beyond this. It not only reveals the state of the system but also explains why the system is in that particular state. Additionally, it provides insights on how to fix or troubleshoot that particular state. This leads us to our next topic: the three pillars of observability.

What are the Three Pillars of Observability?

The three pillars of observability are metrics, logs and traces.

Metrics: A metric is a measured value derived from system performance. It usually has information like memory or latency. For example we might observe a http request is failing over the last 30 min, 10 http request have been failed. You can retrieve this historical data using metrics.

Logs: A log is a record of system event. It is computer generated and timestamped and written into a file which cannot be modified. Continuing with previous example, at 10pm we can check who had sent this http request and determine which part of the application did it hit and why it failed.

Traces: A trace is a record of a series of casually related events on a network. The events do not need to take place within a single application, but they should be a part of the same request flow. As per our example it will be the complete http request tracing. Right from the client hitting the Load Balancer and from there the request went to frontend and then backend and finally database. So using traces we can understand how the request went and how much time it took and the expected time.

Is Observability and Monitoring same ?

In simple terms, monitoring covers one pillar of observability which is metrics. With metrics we can also implement alerts and dashboards and with this we can check what is happening. Whereas observability is a system property that helps you understand why it is happening and how to fix it.

Who is responsible ? Developer or DevOps Engineer ?

To be honest, no one in specific is responsible. It is a collective efforts. Let me break it down.

For an application with infrastructure and networking, the application itself should instrument the logs and metrics. Developers need to instrument these such as metrics or logs such as info or error logs. If they don't instrument, even a well setup Prometheus by a DevOps engineer is use less. As Prometheus will look for metrics and ELK will look for logs. Similarly if tools like Prometheus aren't deployed in kubernetes cluster, the instrumented metrics and logs are of no use. Hence it is a collective effort.

This will be a series of observability articles, where we will cover right from theory to hands-on sessions.

Stay Tuned🔥

If you found this post useful, give it a like👍

Repost♻️

Follow @Bala Vignesh for more such posts 💯🚀