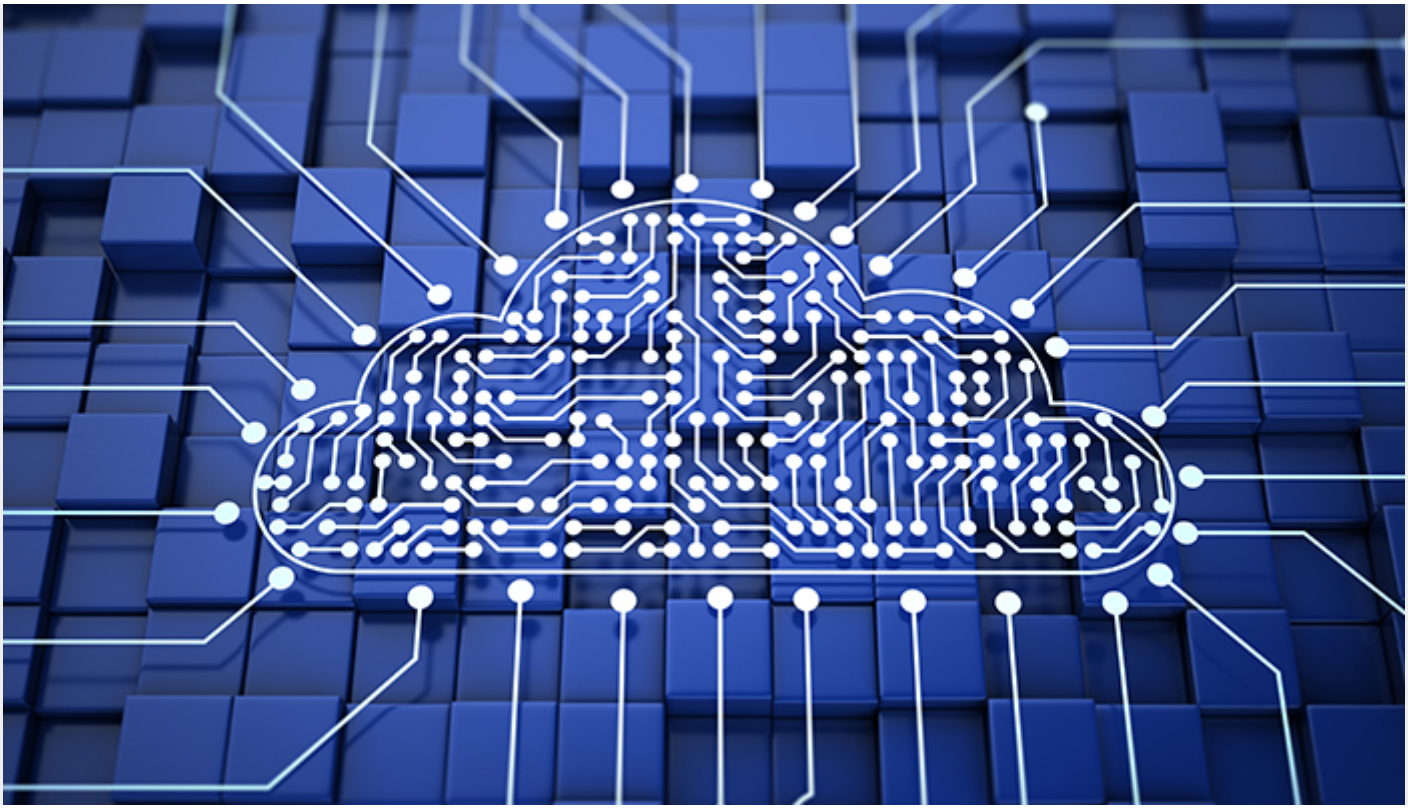
BRINGING INTELLIGENCE TO YOUR OBSERVABILITY PLATFORM – PART 1



I used to be amused by how intolerant my family (non-IT industry) was with slow-to-respond websites or buffering TV programmes. However, after about 30 years in the IT industry, I have relatively recently (over the past five years or so) started to find myself far less forgiving of these situations as well. It's difficult to imagine our day-to-day personal and business lives without well-performing screens, tablets, mobile phones, and other connected devices. Likewise, it is vital that the services we offer our customers are available and perform well when they're needed, which is often 24x7.

Therefore, when something does happen to cause an interruption in service, we need an intelligent observability platform to enable a quick response and initiate the repair. Typically, when an issue occurs with a business service, all the information required to find the root cause is available. The problem is finding that information quickly and taking the appropriate action. There are different elements to an observability platform, and I will look at how the building blocks should be assembled to ensure satisfied customers. In this first post, I will walk through what an observability platform is and what makes it successful.

Observability vs. monitoring

Observability goes beyond mere monitoring (even of very complicated infrastructures) and is instead about building visibility into every layer of your business. Increased visibility gives

stakeholders greater insight into issues and user experience, reducing time spent firefighting and creating time for more strategic initiatives. It's also critical to the overall success of site reliability engineering (SRE) and DevOps organizational models. In modern DevOps development processes, developers need visibility into operational performance as much as traditional ops teams or SREs.

I think of observability as being able to use the vast amount information available from metrics, logs, and traces, including topology, to achieve four things:

1. Understand and display the state of the service
2. Highlight potential issues to avoid a service disruption
3. Identify cause of any service impact or outage
4. Speed resolution to restore service

Historically, the monitoring of an infrastructure and its applications could almost be an afterthought as systems were put into production. There would typically be an expectation/acceptance of issues with any new go-live. It would be the problem of the operations team to manage the business-as-usual (BAU) aspects of the environment.

These days everyone is far less accepting of any performance or availability issues. The DevOps world has moved us forward from big-bang releases to smaller, incremental releases of defined functionality. That should make it simpler to create, test, rollout, observe, and rollback if necessary.

Social media, either public or private, within an organisation makes it easy for unhappy customers to broadcast their views. Often people are not even really venting their frustration; they are just trying to crowdsource an answer or workaround in the expectation that someone else has already done it. Unfortunately, the question on its own can generate a lot of negative sentiment.

High expectations and potential negative publicity mean that an effective and intelligent observability platform is vital to understanding the state of the service and avoiding outages, and when issues occur, fixing them quickly.

Metrics, logs, and traces

The observability platform needs to consume, process, and analyse metrics, logs, and traces to get a full understanding of the health of the business services being provided. These can be natively collected by the platform or consumed via integration with other tools.

Metrics

Today, there is a rich array of metrics available for all components, whether hardware or software. There are also many platforms that provide inbuilt monitoring for multiple different layers of the technology stack: servers, storage, networks, virtualization, containerization, applications, and cloud platforms. Modern applications are built on top of various layers of abstraction/virtualization, which increases the number of metrics. This translates into a vast amount of information that we need to interpret and evaluate so as to maintain the high availability and performance we promise.

I believe it is also critical to collect metrics about end-user performance, which provide the user perspective. A combination of both real and synthetic user activity is ideal. These metrics can be gathered using third-party tools, application-specific tools, or even instrumentation provided by the application. We need to understand which user communities are impacted. Users are not interested in what is inside the black box; they just want the application to work.

Logs

Logs contain a rich amount of data covering the workings or failures of an application or system. Entries are typically timestamped so logs can also give a great indication of how an application is performing even without any reported warnings or errors. Over the years, I have seen a shift from minimal logging, which needs to be increased to “debug mode” in order to troubleshoot an issue, to more detailed logging as the norm. We should no longer need to increase the logging then try to re-create the failure. Ideally, the information should all be there from the initial failure. To maximize the benefits of your observability platform, it needs to be able to consume and analyse large amounts of detailed logs in near real-time.

Traces

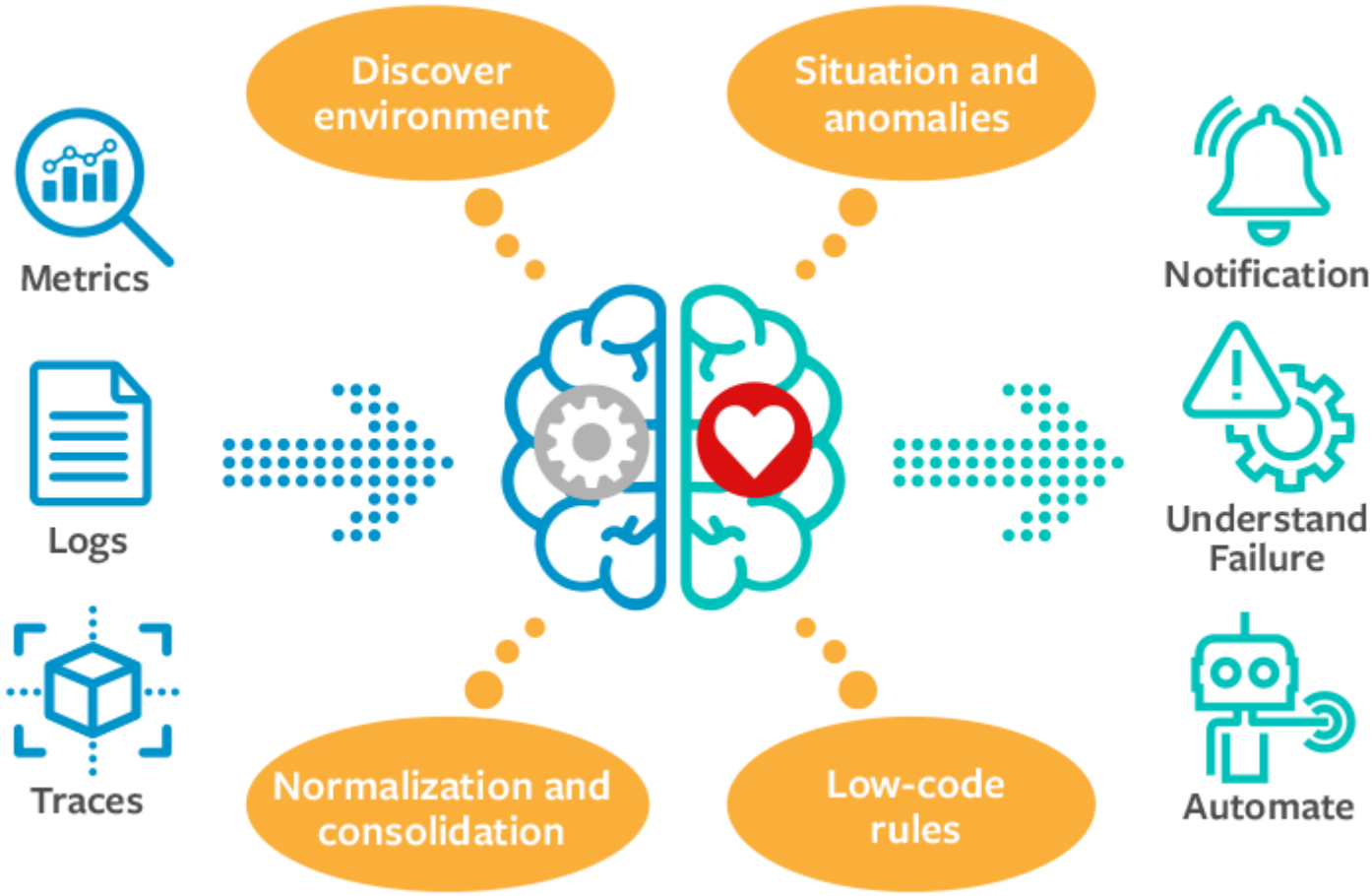
Understanding how end-user transactions are processed by the components of a service provides insight into how different components can impact or rely on each other. In many modern applications, the components used can be very transient, as applications auto-scale to flex with demand from end users. Having an accurate representation of the topology and transaction routes across a period of time is necessary for quick troubleshooting.

Integrations

Integration with many different data sources is key for a successful observability platform. It is virtually impossible for any single vendor to keep up with the rate of change in the wide variety of environments generally used by organizations. Specialized tools for specific environments provide more detail and often support leading-edge features. Open, flexible tool integration with ingested metrics, events, and topology data enables an observability platform to cover the breadth and depth of the most complex environments.

Intelligence

Your observability platform needs to consolidate the various information feeds (metrics, logs, traces, and integrations) to enable automated analysis, utilising all these aspects together with a combination of machine learning and rules.



Intelligent Observability—automated processing of detailed information using artificial intelligence (AI)/machine learning (ML) and rules to resolve issues.

Observability platforms build layers of visibility into your business, giving you the information you need to make decisions when you need to make them. These platforms offer the ability to ingest and analyse metrics, logs, and traces at high volumes, allowing your most important resources to spend their time on more value-adding activities. In my next post, we'll dive into what makes an observability platform successful and the full power the platform can provide.