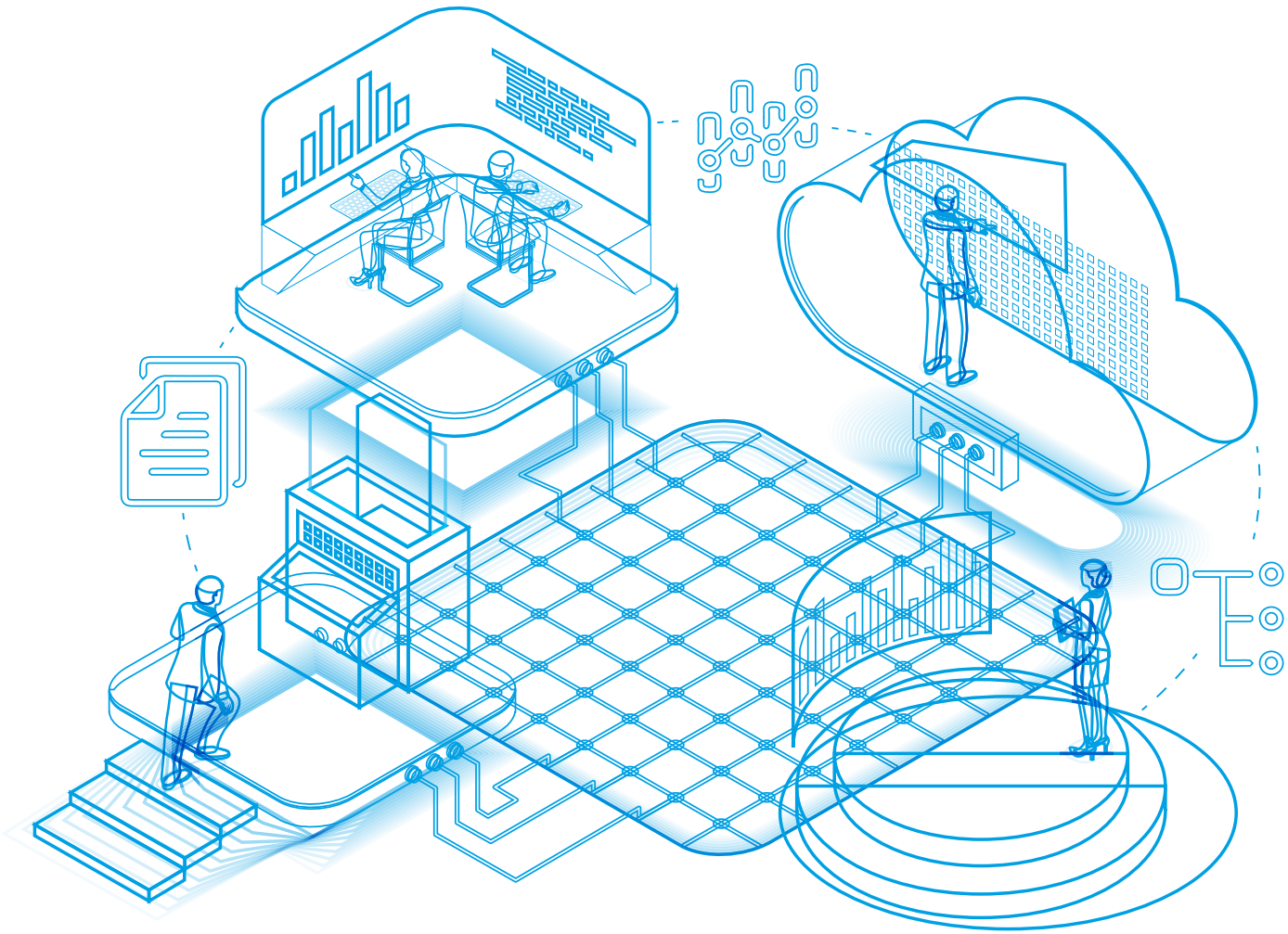
CODIFICATION

Guide to Observability

Understanding Observability



July 2022

Field Guide

Table of Contents

What is Observability?

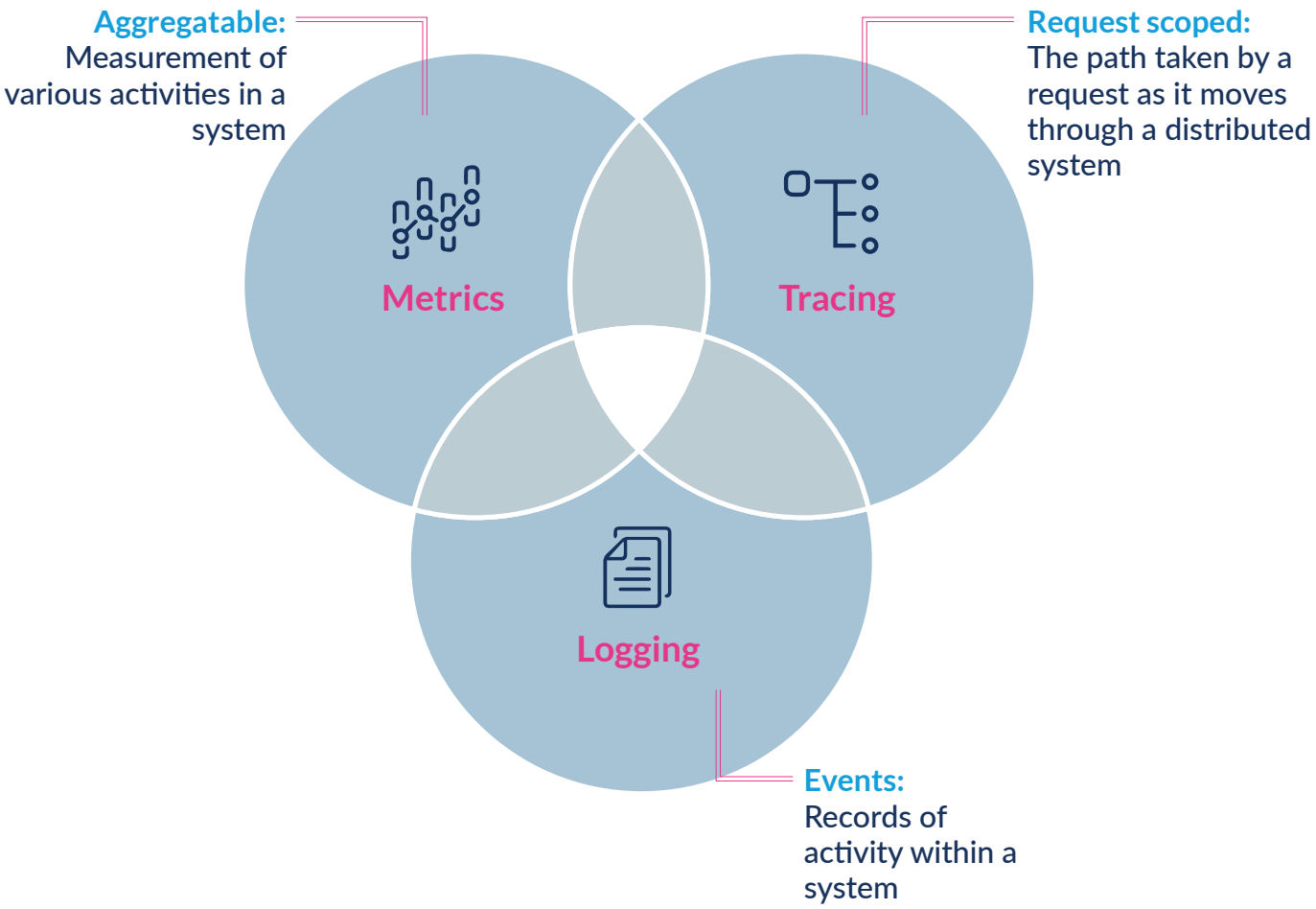
The term “Observability” relates to how much of the internal state or environment of a system can be inferred from the outputs of that system.

It is the ability to gather and analyse the information you need to prevent incidents from happening. The higher the Observability of a system, the more quickly and accurately you can locate and rectify root causes for problems such as performance issues, without the need for additional development or testing.

Observability also involves software tools and practices which are used to analyse performance data from applications and hardware to effectively monitor and debug systems to meet business requirements and customer expectations.

3 Pillars of Observability

Observability does not necessarily mean “monitoring”; rather, it is based on three key pillars known as:

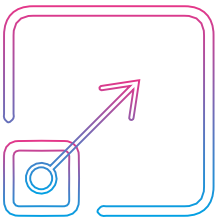


Achieving Observability is important to ensure that there are no service disruptions to meet service-level objectives (SLOs). Businesses need to observe and analyse every aspect of their applications, so any anomaly gets detected and rectified right away.

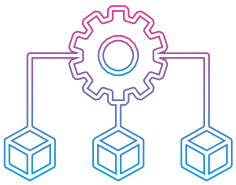
Benefits of Achieving Observability

Discover and address previously unknown issues: with Observability, you will be able to detect issues that weren't known otherwise previously.

Detect issues earlier than usual in development: by monitoring earlier into the development process, you will be able to detect issues earlier.



Scale automatically: through Kubernetes cluster configuration, you will be able to gather data from the onset.

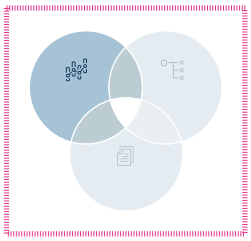


Enable self-healing for application infrastructure: by combining automation and AiOps, you will be able to predict issues by inferring them from system outputs and have them resolved automatically.

You can understand the internal status of your system using signals and output that is apparent to others measured by Observability. The phrase refers to a single cohesive ability. You should be able to see your complete system's status with the aid of Observability.

The Three Pillars: *Metric, Traces, and Logs*

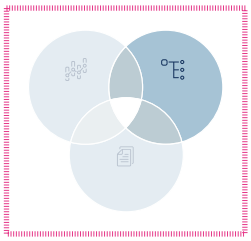
Metrics, Traces, and Logs are the essential data types of Observability. Aggregating data from disparate sources, including logs, gives you the ability to analyse, visualise, and troubleshoot your systems. When you unify these three “pillars” into one cohesive approach, a new ability to understand the full state of your system emerges.



Metrics

The metrics pillar reflects the state of the system as a time series of numbers that are essentially used as measures. Most metrics tools allow you aggregate performance over a few labels. For instance, Prometheus, a metrics monitoring tool, can be used to trigger an alert when the metric exceeds specified thresholds for a predefined length of time.

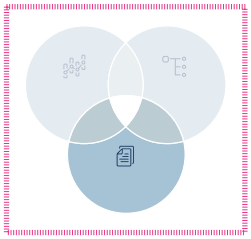
To narrow down your data to a more specific level, you need to see metrics that have high cardinality. This allows you to create views that can visualise the various data points that are related to a particular service. For instance, you can visualise the information that is collected about the users who are actively using a particular service. In terms of representation, metrics are a magnifying glass that can be used to look at the data.



Traces

Traces help you examine individual system calls and understand the underlying procedures that were taken to provide a result. With high-cardinality metrics, you can pin down what is failing and which users are affected.

Traces can be a useful way to more deeply investigate an alert triggered by your metrics system. Traces allow you to see things like which component took the longest or shortest amount of time to execute, or what specific functions did to cause an alert to occur. Tracing tools, such as Jaeger, provide wonderful visibility into what's happening with underlying components.



Logs

Tracing allows you to examine what happened at the system or software level and where the underlying problem occurs. Logs will provide the necessary insights into raw system information that can help us figure out what happened.

In addition to being able to visualise the various data points that are related to a particular service, logs can help you identify issues that are affecting the operation of a database. For instance, if you see that a database is overloaded, you might want to investigate why it is taking too long to load. However, these types of reports are limited by their ability to provide a quick overview.

Some Observability solutions combine the views from various pillars into a single view. However, this method can be very challenging to determine the differences between the different views. For instance, some of the attributes are convenient to see in one view, while others are completely invisible in another. Additionally to being able to visualise the multiple data points that are related to a particular service, logs can also help you identify issues that are affecting the operation of the database.

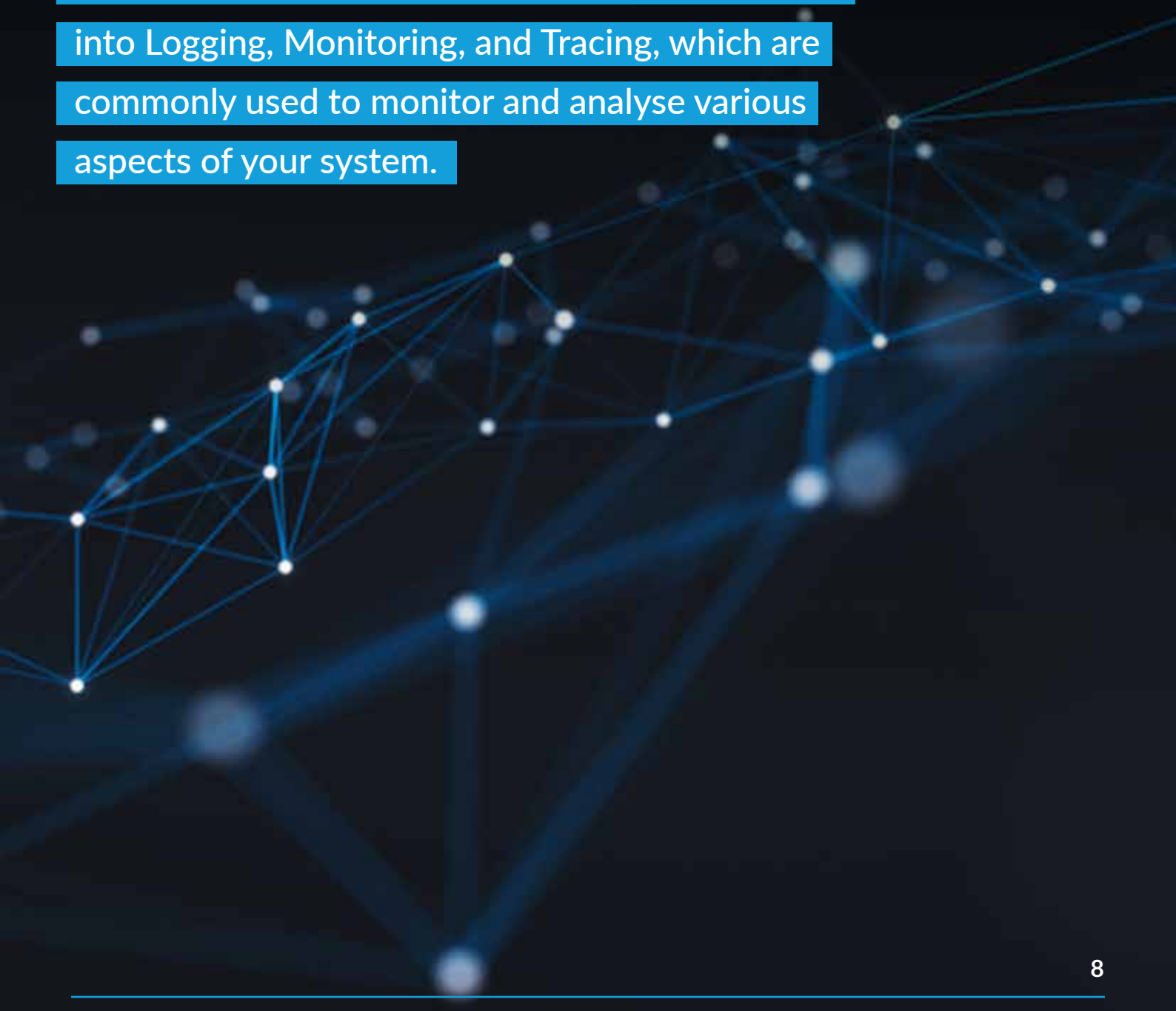
Observability Tool

An Observability tool must be able to keep a single set of data and storage that can offer a rich context in addition to being able to collect and preserve the richness and dimensionality of the data. The events that occur in the data can be visualised using this. For instance, it is simple to notice how two sets of events differ from one another if they have different dimensions.

A unified approach is able to collect and preserve the rich and dimensionality of the data. It allows us to move through the data efficiently and effectively. When implementing a strategy, an important factor to consider is the ability to quickly identify and analyse the various anomalies in the data. With a single tool, we can easily find out where and how often an anomaly occurs in the data.

Cloud Native Observability

You can instantly detect and analyse irregularities in the data through the use of a cloud native computing foundation (CNCF). This method can explore numerous layers and automatically detect and resolve issues. Tools are broken down into Logging, Monitoring, and Tracing, which are commonly used to monitor and analyse various aspects of your system.

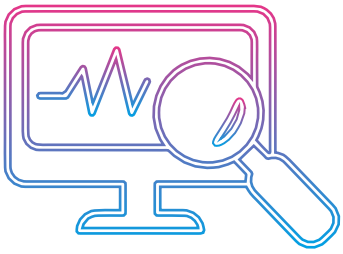


Logging

A critical component of building a modern platform is the collection and analysing of a steady stream of log messages. The emergence of cloud native patterns and the increasing number of containers has changed the way log messages are handled.

Traditional approaches to logging, such as writing logs to a file, are not ideal for applications that rely on a file system for their operations. In a cloud native environment, you need a log-collection tool that runs alongside the containers and collects messages directly from the applications, and sends the data to a central log store to be analysed.

Monitoring

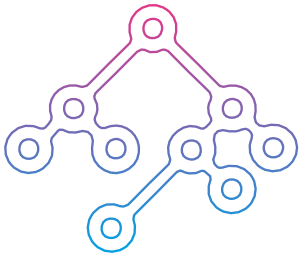


In a cloud native environment, you need to track metrics, logs and events to understand the health of your applications. Monitoring tracks everything from application health to user behaviour and is an essential part of effectively running applications. It helps an application to collect, aggregate, and analyse logs and metrics.

While logs are used to describe the activities of an application, metrics are used to measure the health of the system. A monitoring tool can analyse various aspects of a system, such as its CPU usage, disk space, and memory consumption. It can also perform detailed synthetic transactions to determine if an application is responding securely and correctly.

Tracing

Tracing is a specialised type of logging that can be used to track the path of requests that are sent through a distributed system. You can also use tracing to analyse the health of the system. Having a unique identifier can allow you to follow the transactions that are happening in your system and debug problematic microservices.



Being a powerful tool for debugging a distributed application, Tracing can also help you fine-tune the behaviour of the application. To ensure that the data is collected and analysed in a secure manner, the code must be modified accordingly.

The goal of the Observability and analysis layer is to make sure that your system is secure and productive. Logging tools capture event messages emitted by applications, monitoring watches logs and metrics, and tracing follows the path of individual requests. Together, these tools provide you a complete picture of what's happening within your system.

A Three-Phased Approach

Regardless of whether you are in the banking & financial services, manufacturing, or healthcare and insurance, your customers expect a seamless interaction. It is important that you take the necessary actions to make sure that your applications are operating efficiently. A study found that compared to normal consumers, highly engaged customers spend ninety percent more and generate three times as much revenue annually.

We can define a three-phased Observability maturity framework; reactive, proactive, and data-driven that can be designed to help, to develop and implement effective Observability practices. The goal of these phases is to help engineers and developers understand how changes affect the end user experience and how they can be resolved.

Observability - Reactive Phase

It is imperative that teams prioritise elimination of obvious problems by aggregating as much telemetry data about the system.

a. Instrument and tune.

Your digital leaders and business stakeholders should be involved before you implement a strategy to increase the efficiency of your team. Then, you can keep an eye on the issues that are most crucial to your company.

b. Metrics, events, logs, and traces.

The availability of data types, such as logs, metrics, and traces must be taken into account. It is essential to have the ability to store and analyse these different data types in one location as it allows you to quickly identify and resolve issues in your systems. It is also advantageous since it enables you to gain a greater understanding of your business operations to be able to monitor, visualise, troubleshoot, and analyse various data sources.

c. Establish baselines and set fundamental alerts.

You can uncover areas of concern and issues that could affect the performance of your systems by being able to gather and analyse performance and metrics pertaining to your infrastructure and applications.

To overcome organisational silos between the product, operations, engineering, and service owner teams, service boundary transaction performance must be mapped to product features. You can establish baseline settings and change the sensitivity of your systems as necessary by analysing evident faults. Address your slowest transactions and the most frequent issues to fix your apps. After addressing these issues, you can then start implementing a strategy to increase the effectiveness of your services.

Observability - Proactive Phase

Due to the complexity of your operations, it is hard to identify and address the root causes of issues quickly enough. Setting up SLOs and optimising processes to balance system reliability with speed and address issues before they impact your customers, allows you to improve the efficiency of your services.

a. Set SLOs to align teams.

A service level objective (SLO) is a powerful tool that enables teams to codify their goals and provide clear boundaries on how they should approach their work. It can help them achieve greater freedom and velocity by allowing them to experiment with new approaches. Before you start implementing a new service, make sure that you understand the four golden signals (latency, traffic, errors, and saturation) that can affect service quality.

b. Optimise your alerting strategy.

You can better manage the various aspects of your data aggregation if you have a clear grasp of the team's goals and objectives. A clear alert system can help you respond to problems promptly and increase the effectiveness of your operations. It can also generate a series of straightforward, actionable notifications that can be used to identify problem areas and raise the bar on your service. The majority of businesses concentrate on a core set of measures that indicate when their customers' experiences are deteriorating. This strategy can be carried out through the use of Apdex, a service level indicator that helps DevOps teams align their alerts with the user satisfaction.

c. Improve incident response.

Every minute that you spend analysing and resolving issues related to your data is costing you money. By equipping your operations teams with the necessary automation and intelligence, you can improve the efficiency of your operations and reduce the time it takes to resolve problems.

When establishing an alert system, assigning the responsibility for the health of your services and applications to a team or individual is important. For instance, make sure that incident alerts can be delivered to Slack if your teams are accustomed to communicating with a tool like that. Taking on this kind of responsibility can aid in ensuring that the communication between the different teams is done in a simple and obvious way.

d. Improve incident response.

A fast feedback loop can help reduce the time it takes to resolve issues and improve the reliability of your applications. It helps prevent costly delays in the deployment process. It is important to track the impact of infrastructure changes on their customer experience. This method can help them identify the root cause of their issues and improve their performance.

Observability - Data-Driven Phase

Observability allows teams to take action with their shared data. To achieve this, connect people with the various processes and technology elements of the organisation, and tie it to business outcomes. To keep your data flowing in real-time, automate the instrumentation of new apps and services.

a. SLO trend analysis.

You can find areas to improve the performance of your apps by having a set of key indicators that can be utilised to measure the anticipated behaviour of your applications. By using this technique, you can identify the root cause of your issues and improve the reliability. One of the most effective ways to describe the multiple aspects of a SLO is by using team dashboards.

b. Visualise the relationships among developers, operations, and customers.

A business performance dashboard can help you visualise the performance of your apps and identify areas where you can improve the reliability of your applications. It can also help you triage future work. Having a robust Observability culture can help organisations collect and analyse data, which are often used by business executives to make decisions and improve the efficiency of their operations.

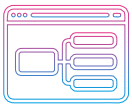
c. Automate instrumentation.

You can implement automation to limit the number of operational tasks. This can help you reduce the time that you spend on manual tasks and improve the efficiency of your operations. The time that you spend on developing apps is usually allocated for the team responsible for the site reliability engineering (SRE). Over time, this group should start to focus on developing only a few operational tasks and will completely focus on the development of the applications. To maximise the efficiency of your operations and prevent you from spending a huge amount of time on manual work, you should implement multiple Observability techniques.

Best Practices on Observability

Collect data:
Gather and examine as much system-related telemetry data as you can, including metrics, events, logs, and traces (whether open source or proprietary), and keep it on a single platform.

Establish a baseline:
Create baseline alert conditions by analysing evident problems, then modify sensitivity as necessary. Create a framework to help you prioritise such actions.

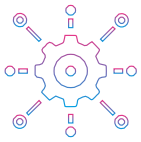


Create SLOs:
Set SLOs using a few pertinent metrics that reflect end-user experiences, maybe beginning with the four golden signals of latency, traffic, errors, and saturation.

Set alerts:
Establish a concise alert strategy that gives priority to a core set of metrics showing a decline in the customer experience.

Speed up the incident report:
Provide your IT operations teams intelligence and automation so they can quickly and proactively identify and address incidents.

Track deployments:
Keep an eye on application deployments so you can gauge how performance may be affected by code modifications. Find the underlying reason for any short-term, long-term, or gradual degradations of your application.



Automate:
Automating instrumentation for new apps and services will keep telemetry data flowing, allowing your teams to move quickly and confidently.

Analyse:
Create dashboards to track SLO progress over time to meet objectives and identify problem areas that require attention.

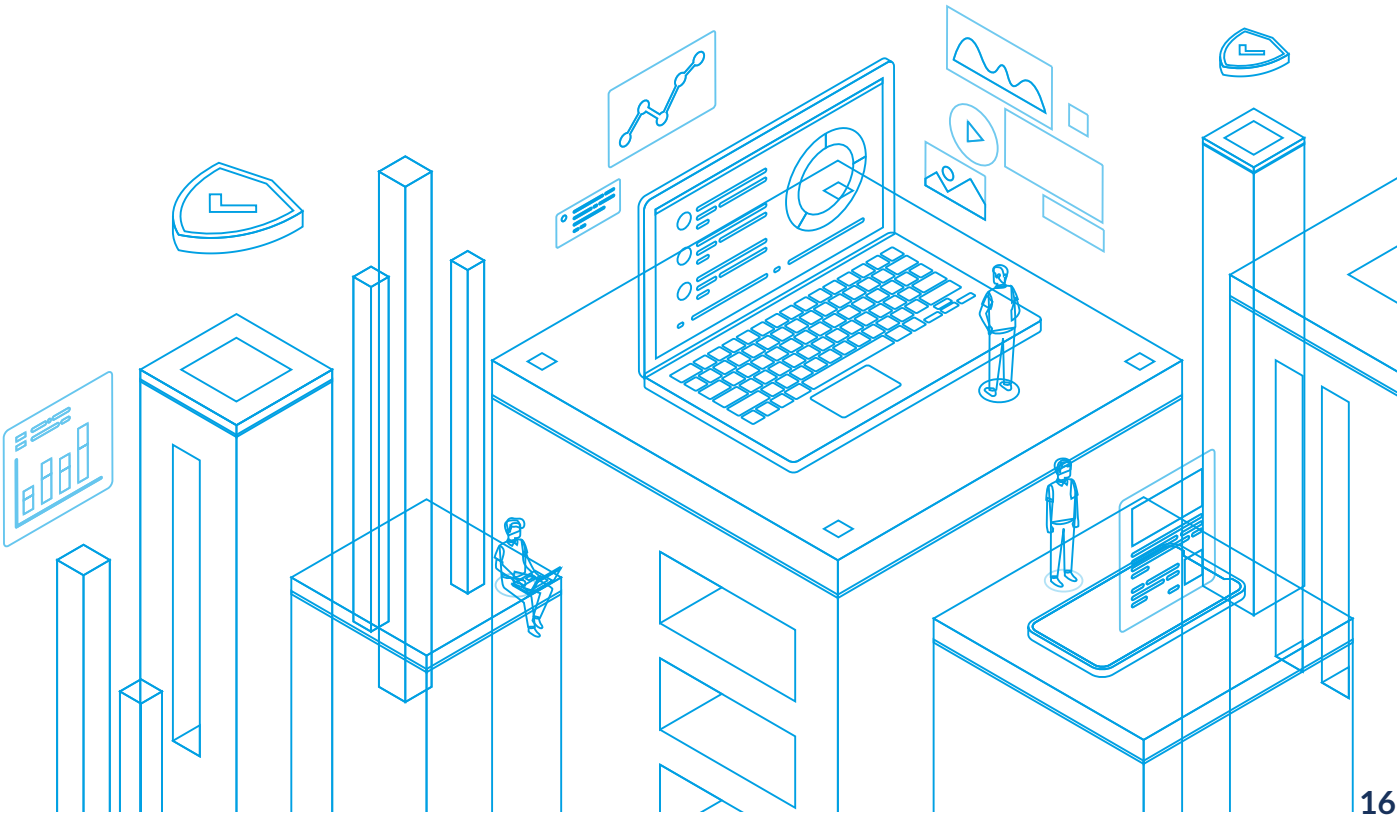
Be customer-centric:
Provide your teams a summary of how customers use or reject your apps.

Conclusion

One of the most critical factors that any DevOps team must consider when it comes to improving the performance and resilience of their systems is Observability. It can help them keep their applications running smoothly and maintain their quality.

As the number of tools and systems that support cloud native applications continues to emerge, the complexity of these systems continues to grow. This is why it is imperative that both the developers and the ops team take the necessary steps to ensure that their applications are running seamlessly.

If your organisation is actively transforming through digitalisation, operational resilience and cyber risk management will be two of the most important elements that must be considered during the entire development process. While enhancing performance and resilience for users and customers, Observability may make sure that the apps and platforms related to it operate securely and effectively.



Looking for more information?

Please visit: www.codification.io

About Codification

Codification is a Cloud Native transformation consultancy, with a team of over 100 engineers, consultants and business professionals distributed across the world. We were founded in 2019 in the United Kingdom. We have grown since then to have a presence in Europe, the Middle East and Asia, serving leading multinational corporations, government institutions, global banks, and industry giants with our consultancy and expertise.

Through our experience, we have noticed that visionary leaders want to transform their organisations into technology companies in order to thrive in the new digital-first economy. Here, businesses want to release software faster, improve quality and build a continuous improvement culture where the best ideas win. At Codification, we establish the direction of a company's technological transformation journey and help implement new technologies and processes, resulting in a modernised digital-ready organisation.

CODIFICATION

Codification United Kingdom
The Core
Bath Lane
Newcastle upon Tyne
NE4 5TF

Phone: +44 01670 223994

Web: www.codification.io/