

# The role of observability in modern businesses

- Logs, metrics, traces ensure system insights
- AI trends enhance anomaly detection
- Real-time monitoring accelerates issue resolution



## Executive Summary

This publication delves into the concept of observability, which is becoming increasingly crucial for organizations. As businesses rely more on interconnected systems and data-driven decision-making, the ability to gain real-time insights into application and infrastructure performance is essential for maintaining operational efficiency and enhancing user experiences.

Key points covered in this ebook include:

- Differences between monitoring and observability,
- Various types of observability.
- Foundational pillars of observability, such as logs, metrics, and traces,
- Challenges organizations face in implementing observability,
- Potential benefits and gains,
- Key trends shaping the future of observability,
- Practical tips for implementing observability,
- Use cases.

This publication offers essential insights for technology professionals, highlighting the transition toward advanced observability practices that improve performance, security, and scalability in organizations.

## Table of Contents

Executive Summary.....	2
Introduction to observability.....	4
The three pillars of observability.....	5
Monitoring vs. observability.....	8
Observability types.....	10
The challenges of observability.....	12
Benefits of observability.....	14
Key trends and predictions in observability.....	16
Tools and solutions.....	18
Tips for implementing observability.....	20
Case studies.....	23
Conclusions.....	26
About CodiLime.....	26

# Introduction to observability

## What is observability?

In IT and cloud computing, observability assesses a system's current state based on the data it produces, particularly logs, metrics, and traces—commonly known as the three pillars of observability. With the increasing complexity of cloud-native environments, observability has become essential. As systems grow more intricate, identifying the root cause of an issue or anomaly becomes increasingly challenging, heightening the risk of significant financial losses for businesses.

Observability is achieved through various tools and approaches that enable teams to detect, analyze, and resolve system issues efficiently. A highly observable system allows teams to quickly and accurately identify the underlying cause of performance issues, often without additional testing, thereby reducing downtime and enhancing system reliability.

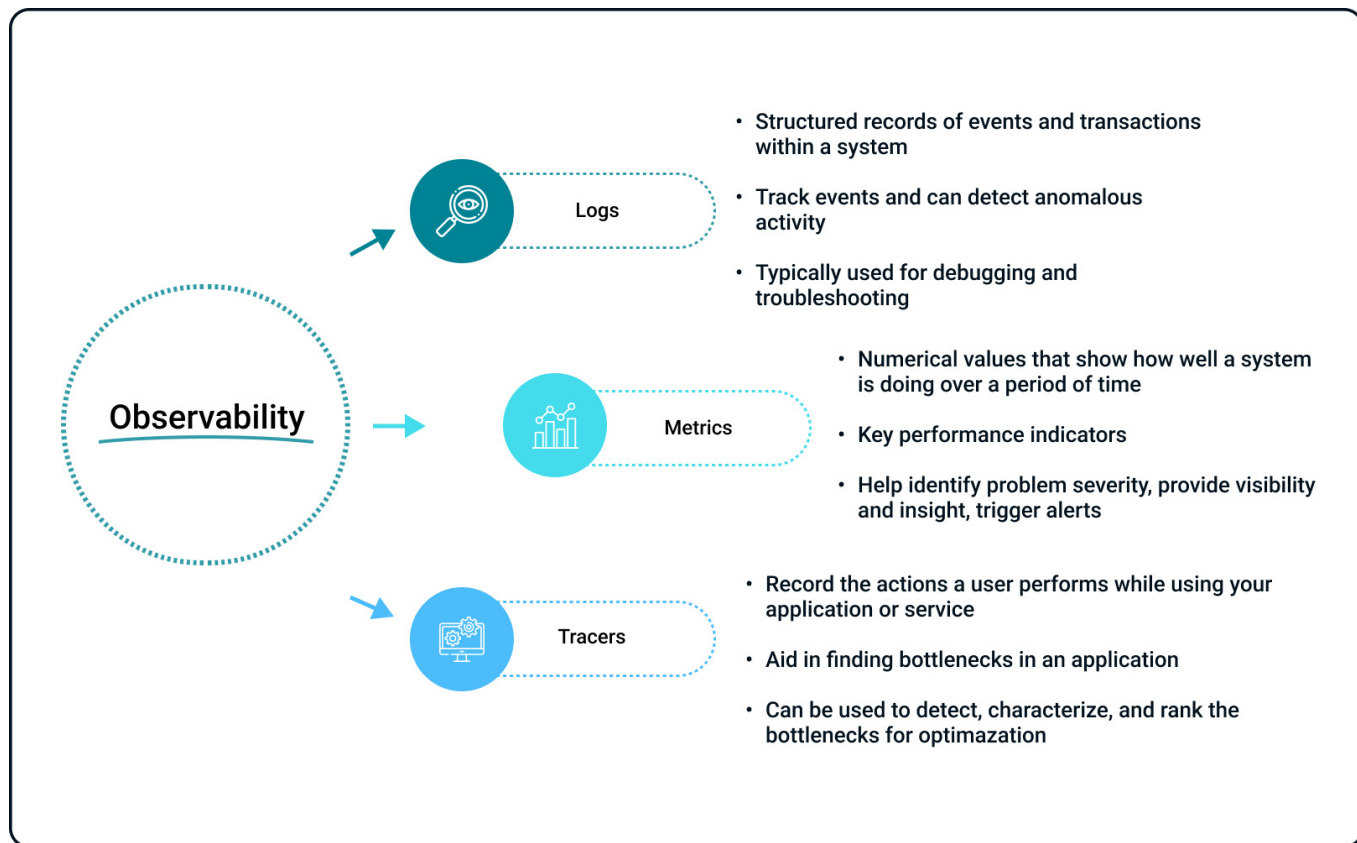
## How observability works

Observability is built on telemetry data from various instrumentation points across endpoints and services in multi-cloud environments. In contemporary systems, each component, be it hardware, software, cloud infrastructure, container, open-source tool, or microservice, generates records of its operations. The purpose of observability is to give teams deep visibility into these diverse environments and technologies, enabling them to quickly identify and address issues to sustain system performance, reliability, and a positive customer experience.

In addition, observability encompasses system-level metrics from the operating system, such as CPU utilization, memory usage, disk I/O, and network statistics, which provide crucial insights into the health and performance of the underlying infrastructure. By integrating these system metrics with application-level data, teams gain a holistic view that aids in proactive monitoring and rapid troubleshooting across all stack layers.

## The three pillars of observability

Logs, metrics, and traces are the key components of observability. Each element has a distinct purpose in capturing various aspects of system performance, and collectively, they establish the groundwork for a robust observability strategy.



### Logs

Every action within your applications produces an event log that includes crucial details such as timestamps, event types, and user or machine identifiers. These are logs—records of discrete events that occur in your system.

Logs include text and metadata information, such as time-stamped information about specific actions, errors, or transactions, necessary for debugging and understanding system events and errors. Therefore, they constitute the first source of information engineers look at when troubleshooting an issue.

Creating logs is relatively easy since most programming languages offer built-in features for event logging. What's more, you can easily incorporate logs into your observability framework with simple modifications.

That said, generating logs can add unnecessary strain to your system, leading to performance

challenges. Such as in high-traffic applications or complex microservices environments. This additional load can impact performance, especially if logs are overly detailed or lack filtering, causing unnecessary data storage and processing demands. Though logs offer deep insights for analysis, it's essential to recognize that system issues are rarely the result of a single event or component. This is why combining event logs with the other pillars of observability is crucial; doing so allows you to gain a more comprehensive understanding of your system's behavior.

## Metrics

Metrics are numerical data points that reflect the performance and health of systems over time. They can be collected, analyzed, and combined to provide a quantitative view of resource usage, performance, and overall system behavior. For instance, organizations can track both current and historical metrics to spot trends and patterns over specific time periods. This approach enables them to define a baseline of normal performance, helping to set benchmarks for future objectives.

Examples of metrics include CPU utilization, memory consumption, request latency, and error rates. Unlike logs, which provide event-specific information, metrics can be used to identify trends and patterns, allowing teams to monitor key performance indicators (KPIs) over time.

Metrics are crucial for real-time monitoring and alerting, helping teams detect anomalies, understand system behavior, and optimize performance by visualizing data in dashboards. They are also effective time-savers, as they can be easily linked across different infrastructure components, providing a complete picture of system health and performance.

## Traces

While logs and metrics are useful for examining the behavior and performance of individual system components, they often fall short in providing insights into the full lifecycle of a request within a distributed system. This is where distributed tracing becomes valuable.

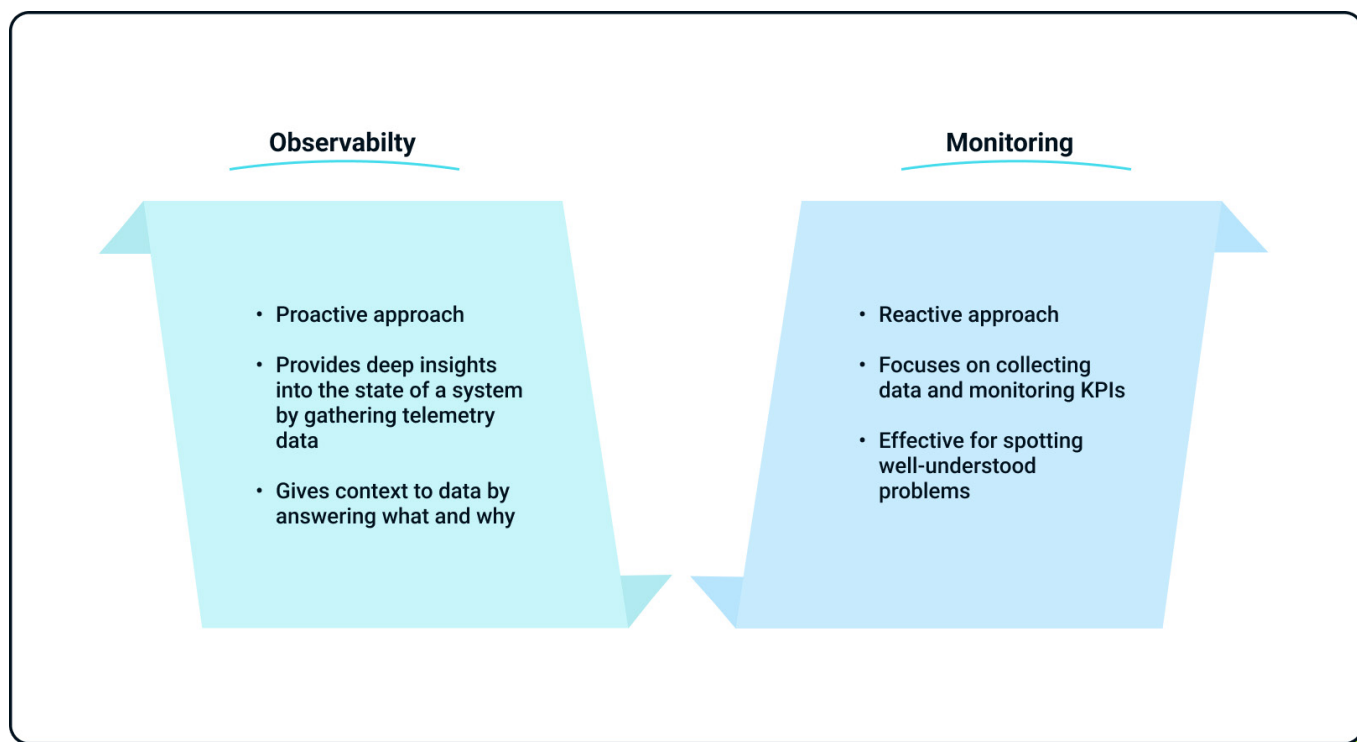
Traces track the flow of a request or transaction as it moves through a distributed system, such as microservices or cloud-native applications. They capture the entire journey of a request across various services, databases, and processes, providing thus a detailed view of how components interact and where delays or failures occur. Each trace is a collection of "spans," which represent individual units of work within a system. By correlating these spans, traces offer insights into the performance of each part of a distributed application. This helps teams identify bottlenecks, track dependencies, and improve overall efficiency.

A significant challenge with tracing, however, is the sheer volume of data generated, particularly in large-scale applications. The quantity of traces can quickly become overwhelming, complicating efforts to analyze and extract actionable insights. As a result, traces are often sampled, with only a subset retained to manage data volume effectively.

Together, logs, metrics, and traces provide a comprehensive, multi-dimensional view of system behavior. Logs offer detailed, event-specific insights; metrics provide a high-level performance overview; and traces help to reveal end-to-end application flows. This combination makes detecting, diagnosing, and resolving issues in modern IT environments easier.

## Monitoring vs. observability

While the terms “monitoring” and “observability” are often used interchangeably, they represent distinct concepts. Understanding the difference between the two is key to building a solid and proactive approach and managing modern distributed systems.



### Monitoring

Monitoring refers to the process of tracking predefined metrics, events, and logs to assess the health and performance of systems. Monitoring tools use dashboards and alerts to inform teams when key metrics deviate from established baselines. For example, monitoring can notify teams if CPU usage spikes above 90%, a server drops, or becomes unavailable. Monitoring is reactive, focusing on detecting known issues based on thresholds or predefined events. It is effective for spotting well-understood problems but can fall short when it comes to detecting complex or unexpected issues that lie outside predefined parameters.

### Observability

Observability, on the other hand, is a broader and more proactive approach. It goes beyond simply tracking known metrics and events. It aims to provide deep insights into the internal state of a system by collecting and analyzing telemetry data such as logs, metrics, and traces. Observability



enables teams to answer not just what is happening but also why it's happening, even in cases where the issue has not been previously defined. It enables teams to troubleshoot complex issues, and gain a deeper understanding of system behavior.

Unlike monitoring, which focuses on tracking predefined metrics within individual systems, observability actively uses data to provide a comprehensive view of the entire technology stack. It consolidates data from all IT systems to generate real-time insights, detect anomalies, identify their root cause, and address them proactively.

While observability traditionally focuses on logs, metrics, and traces, modern cloud-native environments require a broader scope. To achieve full observability in these complex setups, it's essential to also consider metadata, user interactions, network and topology insights, and access to code-level information.

## Observability types

Observability encompasses various approaches that address specific domains and challenges. In this section, you will find the key types of observability.

### Infrastructure observability

Infrastructure observability spans hybrid environments, containers, virtual machines, servers, storage systems, and more. Below are examples illustrating the crucial role observability plays in some of these components.

- Cloud environments - Since cloud environments are dynamic and scalable in nature, observability enables teams to identify and optimize underutilized resources for cost savings. What's more, it supports performance stability and facilitates rapid troubleshooting.
- Hybrid environments - For hybrid infrastructures that blend on-premises and cloud resources, observability enables unified monitoring by providing a centralized view that consolidates data from both environments. It also supports consistent performance management.
- Containerized architectures - With container technologies like Kubernetes, observability is crucial for dynamic resource management, allowing real-time monitoring and adapting to workload changes. It also provides visibility into microservices interactions for efficient root cause analysis.
- Virtual machines (VMs) and servers - Observability for VMs and physical servers focuses on tracking key metrics like CPU, memory, disk I/O, and network traffic to keep systems running efficiently. It also helps teams quickly spot and fix failures or performance bottlenecks at the server level.
- Storage - Observability covers storage systems by tracking utilization and performance metrics to help prevent bottlenecks.

### Network observability

Network observability is the ability to monitor and understand what's happening within a network in real-time. By collecting and analyzing data from various network sources (physical or virtualized), it reveals how network performance impacts applications, user experience, and business goals. Network observability can be also seen as a part of infrastructure observability.

## Data observability

Data observability is the capability to monitor, assess, and maintain data health across its entire lifecycle. At this level, data and analytics engineers are tasked with designing and managing data pipelines to ensure data remains timely, complete, and of high quality. They work to guarantee that data meets essential standards, enabling reliable insights and smooth operations across systems.

## Machine learning observability

ML observability centers on monitoring and evaluating model performance, specifically assessing how accurately models can predict potential system failures. Machine learning engineers and data scientists rely on ML observability to gain deeper insights into model behavior and effectiveness throughout each stage of development.

## Application observability

Application observability is tracking and understanding how an application performs by collecting data like logs, metrics, and traces. It helps teams spot issues, slowdowns, or errors in real time so that they can fix problems quickly. With tools that monitor the app's performance, teams can identify bottlenecks, failures, and performance drops. The goal is to have clear visibility into the app's health and behavior, so problems can be caught early and resolved before they affect users. Application observability integrates with other monitoring systems to provide a full picture of an application's environment.

## Security observability

Security observability offers a comprehensive view of an organization's IT environment, enhancing the ability to proactively identify and address vulnerabilities. It allows for the immediate detection of anomalies and unusual behavior, facilitating faster threat response. By aggregating and analyzing data from multiple sources, such as telemetry and logs, these tools help identify potential security issues early and can automate responses to mitigate threats in real time. This approach goes beyond merely detecting known threats, providing a detailed blueprint of the security landscape to minimize blind spots and speed up reaction time.

## The challenges of observability

Observability is essential for understanding the health and performance of complex systems, but achieving it comes with significant challenges. In this section, we will explore the major hurdles that organizations face when striving for effective observability.

### Infrastructure complexity

Managing observability in dynamic multi-cloud environments is challenging due to the complexity of tracking real-time changes across different cloud platforms and services. Consolidating data from various cloud providers and ensuring efficient cross-platform visibility often requires specialized solutions.

### Microservices complexity

In microservices architectures, observability can be even more challenging due to the distributed nature of services and containers. Tracking the interactions between numerous microservices, each with its own set of dependencies, requires tools that can provide end-to-end visibility and real-time monitoring across these dynamic components.

### Data silos

One of the most common challenges in observability is the presence of data silos. Different teams often use separate tools and systems to collect, store, and analyze data. This fragmentation leads to a lack of unified insights, making it difficult to get a complete view of system performance. When observability data is scattered across silos, it becomes nearly impossible to correlate metrics, traces, and logs, resulting in inefficient operations and missed opportunities for optimization.

### Data volume and speed

The massive scale, speed, and variety of data and alerts make observability difficult to manage. For instance, high-traffic messaging systems or real-time analytics platforms can generate billions of data points that require great effort to organize, filter for errors, and notify the appropriate developers.

### Manual instrumentation and configuration

Setting up observability often requires manual instrumentation and configuration, which is not only

time-consuming but also prone to human error. Developers must manually add monitoring code into applications, leading to inconsistencies and gaps in coverage. This burden distracts teams from core development tasks, increasing the time to market for new features. Manual processes also make scaling observability efforts more difficult as systems evolve and grow.

## **Multiple tools and vendors**

Many organizations rely on a patchwork of tools from different vendors for observability. While each tool may provide valuable insights, managing and integrating multiple platforms adds complexity to the process. Switching between various tools to gather metrics, logs, and traces wastes time and resources and the lack of a unified solution complicates troubleshooting and performance optimization.

## **Cost of observability**

Collecting and processing large amounts of data can lead to substantial expenses, especially when scaling across large, distributed systems. Moreover, using multiple tools and vendors not only increases complexity but also inflates costs. Balancing the need for comprehensive observability with budget constraints is a difficult challenge for many organizations.

## Benefits of observability

Maintaining the health and performance of complex systems is more crucial than ever. Observability goes beyond traditional monitoring by providing a deeper, data-driven understanding of systems. It empowers organizations to detect, resolve, and prevent issues before they impact end users. In this section, we'll explore the key benefits of observability and how it can transform your operations.

### Proactive issue discovery

One of the most significant advantages of observability is its ability to discover issues that you didn't even know existed—also known as “unknown unknowns.” Traditional monitoring tools are often limited to predefined thresholds and alerts, which means that they can only catch problems you've anticipated. However, modern systems are too complex and dynamic for such a limited approach.

With observability, you gain full visibility into the behavior of your system through metrics, logs, and traces, which allows you to discover unusual patterns or anomalies that don't fit typical failure scenarios. This proactive issue discovery helps teams resolve problems before they evolve into significant failures.

### Early issue detection and resolution in development

Observability isn't just for production environments; it also plays a crucial role during the development phase. Incorporating observability into development and pre-production environments allows teams to detect performance bottlenecks, misconfigurations, and other issues before they make it into production.

By catching these issues early, development teams can significantly reduce the cost and effort of fixing them. Early detection prevents expensive rework, limits the risk of deploying buggy software, and improves time to market for new features. Moreover, it enhances collaboration between development and operations teams by providing shared visibility into how systems behave across the entire software delivery lifecycle. This integrated approach helps organizations ship better, more reliable software faster.

### Automated scalability

Modern systems are often designed to be highly dynamic, scaling up and down automatically based on demand. However, without proper observability in place, managing this scalability can become complex and error-prone. Observability provides real-time insights into the system's performance and resource utilization and allows automated scaling to occur more efficiently.

By continuously tracking performance metrics, such as CPU usage, memory consumption, and network latency, observability tools can trigger scaling actions when thresholds are met. For example, an observability platform can automatically spin up more instances when traffic surges or scale down resources during off-peak hours. This not only optimizes the performance and availability of your application but also helps control costs by ensuring resources are only used when necessary.

With automated scalability, observability eliminates the need for manual interventions and fine-tuning, allowing businesses to handle changing demands seamlessly.

## **Automated remediation and self-healing**

Another significant benefit of observability is its ability to support automated remediation and self-healing systems. Observability provides a rich set of telemetry data that can be used to identify and resolve issues automatically. For instance, if a system detects abnormal behavior—such as increased latency, failed transactions, or resource exhaustion—observability tools can trigger predefined workflows that mitigate the issue without human intervention.

These workflows could include restarting services, rerouting traffic, scaling up resources, or even rolling back faulty deployments. With automated remediation, systems can “self-heal” in real-time, reducing downtime and improving system reliability. This not only frees up valuable time for IT and DevOps teams but also ensures faster response times and enhanced system resilience.

Automated remediation, combined with observability, paves the way for more autonomous and reliable infrastructure, where issues are detected and resolved instantly, often before end users are even aware of a problem.

## **Enhanced mean time to resolution (MTTR)**

Rapid detection and resolution of issues are essential in limiting downtime. Observability tools play a key role in lowering MTTR, enabling faster problem-solving, and reducing the costs tied to system outages.

## Key trends and predictions in observability

As observability evolves, several important trends are emerging due to the growing complexity of systems, economic considerations, and technological progress. Here are the key trends and predictions in the field of observability:

### Correlating operational and business data

Organizations are increasingly looking to bridge the gap between operational performance data and business outcomes. By correlating operational metrics with business key performance indicators (KPIs), observability tools help teams understand how system performance impacts customer experience, revenue, and overall business health. This shift allows businesses to make more informed, data-driven decisions and prioritize issues that have the greatest impact on their bottom line.

### Tool consolidation

As enterprises adopt multiple observability solutions to address different parts of their stack, such as application performance, infrastructure monitoring, and security, there is a growing demand for tool consolidation. Companies are seeking unified platforms that can provide end-to-end observability across all layers of their systems, reducing complexity, costs, and silos between teams. The push for consolidation is also driven by the need for seamless integration, easier data management, and a unified user experience.

### Increased adoption of cloud-native technologies

With the continued shift toward cloud-native architectures, including containers, microservices, and serverless computing, the demand for observability solutions that can handle these dynamic environments is increasing. Cloud-native observability tools are designed to monitor highly distributed and ephemeral systems in order to provide real-time visibility into performance, dependencies, and resource utilization. This trend will drive further innovation in observability platforms, enabling better support for hybrid and multi-cloud environments.

### Advancement in machine learning and analytics

Machine learning (ML) and advanced analytics are playing a growing role in observability. These technologies are enhancing capabilities in anomaly detection, root cause analysis, and predictive maintenance. By leveraging AI/ML algorithms, observability tools can automatically detect patterns in vast amounts of telemetry data and surface potential issues before they escalate.



## Pushback on traditional pricing models

As observability platforms have grown in complexity and feature sets, some organizations are pushing back against traditional usage-based or per-host pricing models, which can become prohibitively expensive at scale. This has led to a demand for more flexible and transparent pricing structures that better align with the actual value delivered by the tools. Vendors are responding by exploring new pricing models, such as value-based pricing or fixed-rate subscriptions, to accommodate customer needs and retain competitiveness in the market.

These key trends and predictions highlight the ongoing transformation of observability. Organizations are seeking more efficient, cost-effective, and comprehensive solutions to handle the increasing complexity of their systems, driving technological advancements.

## Tools and solutions

Monitoring and observability tools play a critical role in ensuring the performance, reliability, and security of applications and infrastructure. These tools provide deep insights into system behavior, allowing teams to detect and resolve issues before they impact users. Below are some of the leading tools and solutions in the market:

### AppDynamics

AppDynamics is an application performance management (APM) tool that provides real-time monitoring of applications and infrastructure. It supports the following capabilities: Infrastructure Monitoring, Log Observer Connect, Network Explorer, Synthetic Uptime Monitoring, APM (incl. Always On Profiling), Synthetic API Monitoring, Real User Monitoring, and Synthetic Browser Monitoring.

### Datadog

Datadog is a cloud-native monitoring and security platform that covers a wide range of use cases, including infrastructure monitoring, log management, APM, and security monitoring. Datadog offers more than 800 built-in integrations across various systems, apps, and services.

### Dynatrace

Dynatrace is an AI-powered observability platform offering full-stack monitoring and automated root-cause analysis, ideal for large enterprises with complex, cloud-native environments.

### Grafana

Grafana offers a comprehensive observability stack that includes tools for monitoring and visualizing application, infrastructure, and cloud performance. At its core is Grafana Cloud, an observability platform that integrates numerous Grafana open-source products and provides additional features available exclusively in its paid version, such as synthetic monitoring. With integrated AI/ML capabilities, automated anomaly detection, and contextual root cause analysis, Grafana helps teams quickly identify and resolve issues. In addition, Grafana open-source solutions like Grafana Loki for logs, Grafana Tempo for tracing, and Grafana Mimir for metrics work together to provide real-time insights across various data sources. Additionally, it supports load testing with Grafana k6, synthetic monitoring, and incident response through Grafana OnCall, making it a versatile solution for various observability needs.

## Kentik

The Kentik Network Observability Platform is a SaaS tool for monitoring networks across data centers, clouds, and containers. It helps troubleshoot issues, detect anomalies, protect against DDoS attacks, and manage costs with real-time data and dashboards. Features include WAN/SD-WAN monitoring, Kubernetes tracking, and synthetic testing.

## New Relic

New Relic is a full-stack observability platform that monitors applications, infrastructure, and cloud services, offering tools like APM 360, real user monitoring, and AI-powered insights. It simplifies performance optimization with unified telemetry, advanced logging, and features like alerts, dashboards, and change tracking.

## Selector AI

Selector AI simplifies performance management and troubleshooting for applications, networks, and cloud infrastructure. It consolidates data from multiple sources, using machine learning to normalize and correlate metrics, reducing repair times by up to 90%. With natural language queries and integrations like Slack, teams can quickly access insights and resolve issues efficiently.

## Splunk

Splunk is a unified platform for security, observability, and data analytics, designed to help teams ensure reliable and secure digital systems. It offers tools like Enterprise Security, SOAR, and Application Performance Monitoring, along with AI-powered insights and over 1,000 data integrations.

## Tips for implementing observability

This chapter provides key strategies for implementing observability in your organization. From defining objectives and choosing the right tools to setting up effective data collection and monitoring practices, these tips will help you achieve effective observability and improve your system's performance and reliability.

### Tips for implementing a continuous improvement approach

Adopt a clear vision and set measurable goals



Encourage employees at all levels to participate



Use the right tools



Evaluate progress and adjust strategy



Identify the main challenges and obstacles



### Establish clear goals

Start by clearly defining the specific objectives for implementing observability in your organization. Articulate the primary outcomes intended from observability across applications, infrastructure, and networks, and identify the benchmarks to be achieved along with the criteria that will measure success. Ensure these initiatives align with business-oriented Service Level Objectives (SLOs).

Clarify aspirations, such as minimizing downtime, enhancing performance, or improving user satisfaction.

## Define observability requirements

Determine the types of data to collect for observability. This involves identifying relevant logs, metrics, and traces and categorizing them for better visibility. Your requirements will vary based on your system type; for instance, you may need application logs and infrastructure metrics. The level of detail required will depend on your goals—granular traces for latency issues or broader metrics for resource optimization.

## Choose the right tools and technologies

Selecting the appropriate observability tools is crucial for meeting your organization's specific requirements. Take into consideration the following aspects: scalability, ease of integration, customization options, and automation capabilities. Always make sure to choose a platform that can grow and evolve alongside the organization's needs.

## Ensure the platform is future-ready

Select observability tools and platforms that are adaptable and scalable to meet future demands. As your organization evolves, the observability system should be capable of integrating new data sources and functionalities. This forward-thinking approach will help maintain robust observability practices as your infrastructure grows and changes over time.

## Centralize and automate data collection

Implement a centralized data collection system that automates the gathering of logs, metrics, and traces from various sources. This will reduce manual effort and ensure consistent and comprehensive data collection across your entire infrastructure.

## Implement near real-time monitoring

Set up a monitoring system that offers near real-time visibility into your application's performance and infrastructure. This allows for quick detection and response to issues, reducing downtime and improving user experience. Near real-time monitoring helps quickly identify any anomalies or performance drops, thus enabling prompt fixes and proactive management. Adding alert systems to this framework can boost responsiveness, notifying your teams immediately of critical incidents.

## Consider data analysis and visualization

Integrate analytical tools capable of efficiently processing and visualizing the data you have

gathered. Your dashboards and reports should deliver timely, actionable insights for your teams. Consider which types of dashboards are most suitable for your specific use case and determine the key metrics you want to monitor. Additionally, assign responsibility for the upkeep of these dashboards to ensure they remain relevant and useful.

## **Ensure proper training**

It's essential to provide your teams with the necessary training to use observability tools efficiently. Promote collaboration among development, operations, and security teams to fully capitalize on the advantages of these tools, whether they are newly introduced or being adapted for application observability purposes.

## **Observability culture**

An observability culture emphasizes proactive monitoring, transparency, and data-driven decision-making across all levels of an organization's technology ecosystem. It encourages teams to move beyond reactive troubleshooting, fostering a mindset that values continuous visibility into system performance and user experience. Engineers, developers, and operations teams collaborate closely in an observability-focused culture, leveraging real-time metrics, logs, and traces to understand and anticipate issues before they impact users. This culture improves system reliability and resilience and promotes accountability and innovation, as teams use insights from observability data to optimize applications, improve processes, and enhance customer satisfaction.

## Case studies

This section contains case studies that highlight CodiLime's innovative solutions for enhancing network management and performance monitoring. These case studies demonstrate how CodiLime's approach to reliability, user-centered design, and automation delivers measurable improvements for our clients across different industries.

### Case study #1 Reliability-as-code solution to provide a proactive approach to network management

network reliability SLO monitoring

CodiLime developed a reliability-as-code solution to enable service providers to proactively manage network performance and prevent outages, ensuring consistent adherence to service-level objectives (SLOs).

#### Challenge:

The client faced difficulties in maintaining consistent service performance and avoiding outages leading to SLO violations. They required a proactive system to monitor and control network behavior dynamically.

#### Benefits:

- Improved service performance through real-time telemetry analysis.
- Reduced downtime with automated flow control and dynamic policy activation.
- Enhanced network reliability, ensuring consistent adherence to SLOs.

#### Solution:

We implemented a reliability-as-code solution that continuously collects telemetry and analyzes network traffic patterns to monitor performance.

Key features included:

- Automatic activation of flow control policies in response to SLO violations, preventing service outages.
- Policies designed as "automatic control circuits" to enable dynamic reactions to various network behaviors.

**Technologies we used:** Python, Golang

## Case study #2 A data visualization portal for an AI-powered infrastructure monitoring platform

infrastructure monitoring data visualization UX design

CodiLime enhanced an AI-powered infrastructure monitoring platform by designing a user-friendly, visually engaging data visualization portal to improve decision-making.

### Challenge:

The client's existing monitoring platform struggled to present data in an accessible and actionable format, making it difficult for users to make informed decisions quickly.

### Benefits:

- Improved data visibility for faster and more informed decision-making.
- Enhanced user satisfaction through a more intuitive and visually engaging interface.
- Increased platform adoption and efficiency thanks to a user-centered design approach.

### Solution:

We developed a comprehensive data visualization portal with the following steps:

- Conducted thorough user research to understand requirements and pain points.
- Designed and iteratively refined a clickable UX prototype for maximum usability.
- Delivered a polished final design that combined functionality with aesthetic appeal.

**Technologies used:** React, MobX, TypeScript, nest, Ant Design

## Case study #3 Automated cloud onboarding for network performance management

cloud integration network performance

CodiLime simplified and automated the onboarding of client systems into a cloud-based Network Performance Management (NPM) solution, improving scalability and observability.

### Challenge:

The client needed to integrate their platform with AWS, Azure, and GCP environments while enabling seamless onboarding of new virtual private clouds (VPCs). Manual integration processes were time-consuming and error-prone.

### Benefits:

- Faster and more reliable onboarding of new VPCs, improving scalability.



- Enhanced network performance management through streamlined integration processes.
- Reduced manual effort, enabling the client's team to focus on strategic initiatives.

**Solution:**

We automated the onboarding process with the following:

- Configured AWS, Azure, and GCP environments, including setting up flow logs to feed into the client's platform.
- Developed Terraform and Ansible modules to automate the integration of new VPCs with monitoring and observability platforms.

**Technologies used:** Python, Go, Docker, Terraform, Ansible

## Conclusions

Observability is a critical component for organizations striving to enhance their operational efficiency and application performance. By providing deep insights into system behavior, observability enables teams to identify and resolve issues before they escalate proactively. This, in turn, results in improved decision-making, minimized downtime, and a better user experience.

## About CodiLime

Since 2011, CodiLime has been the engineering partner of choice for semiconductor companies, networking vendors, telecom services, and software solution providers.

We're home to 250 top-notch software developers, network engineers, DevOps experts, and solution architects. We appreciate long-term collaborations above all, as well as our partners. Below are some names that have already trusted us:



CodiLime aims to link network engineering talent with business domain expertise – we focus on five N.E.E.D.S. - Networks, Equipment, Environment, Data and Security.