devon

Observability At Enterprise Scale: Beyond Monitoring

To be a trusted partner for professionals and organizations to accelerate their journey to high performance

Contents

Abstract

This whitepaper aims to provide you with a deep understanding of Observability and how it can provide insights into the internal health of distributed systems at Enterprise scale. Implementing Observability leads to higher business value, enhanced stability and reliability of services. Observability is not just for Ops but for anyone who needs visibility into how their services are performing in Production.

Introduction

Observability (or O11y) is a concept from control theory that has become increasingly important in recent years as systems have become more complex and distributed. It builds upon modern practices of DevOps, SRE and Cloud native movements. It is regarded as the next generation of monitoring (or "monitoring on steroids"). Investing in Observability is not just a best practice but a necessity for staying competitive and to deliver better experiences for users.

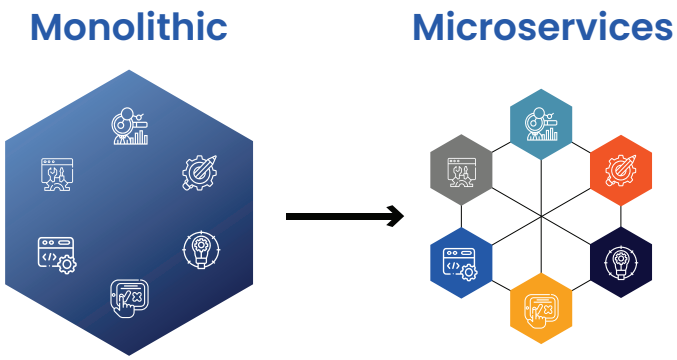
Target Audience

Observability is a multi-disciplinary topic. Target audience for this paper may include below roles:

- Site Reliability Engineers
- DevOps
- Sysadmins
- Software Engineers
- Infrastructure Engineers
- Software Developers
- Product Managers

Why do we need Observability?

In today's world, software is evolving at a tremendous pace. Systems have become more complex. Monoliths are being replaced with microservices. There is an aggressive drive to modernize. Workloads have changed. Infrastructure is not treated and cajoled like a pet, but rather treated as cattle. Services are more dynamic.



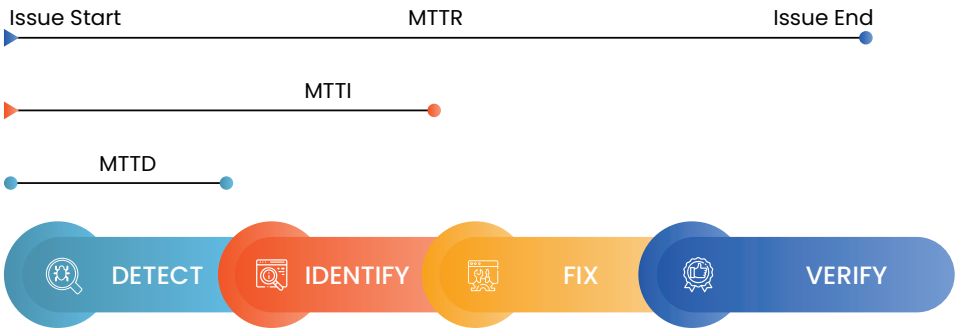
Distributed systems like cloud native applications are never up; they exist in a constant state of partially degraded service. The number of nines in Uptime doesn't matter anymore if users are not happy. Latency is the new down.

The Myth of the Nines

Availability %	Downtime per Year	Downtime per Month	Downtime per Week
99.9% (three nines)	8.76 hours	43.2 minutes	10.1 minutes
99.95%	4.38 hours	21.56 minutes	5.04 minutes
99.99% (four nines)	52.6 minutes	4.32 minutes	1.01 minutes
99.999% (five nines)	5.26 minutes	25.9 seconds	6.05 seconds
99.9999% (six nines)	31.5 seconds	2.59 seconds	.0605 seconds

- Average polling interval for monitoring - 5 minutes
- Even superhuman operations people can't be alerted and take action in under 5 minutes.
- One outage per year could drop service level to three nines or worse.

According to the New Relic 2023 Observability Forecast report, a median annual outage costs around $7.75 million. Critical business application outages now cost more than $500K per hour of downtime. "Troubleshooting tax" – 73% of companies spend at least half of their time troubleshooting / debugging for microservice architectures with every outage seeming to be like a murder mystery! Large scale IT outages have become more prevalent in recent times. Downtime is a disaster for businesses of any size. Even large enterprises like Microsoft, Google, Amazon etc. are not immune from the effects of outages. Incident resolution is invariably expensive. Mean time to recovery (MTTR) and mean time between failures (MTBF) are important metrics in reliability engineering and are used to assess the performance of systems. A low MTTR and a high MTBF are generally desirable.



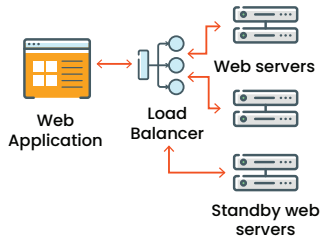
It is becoming extremely hard to troubleshoot failures in microservice architectures. There are component failures E.g. Gray failures which are, by definition, hard to detect, especially from a single point of view.

The eight fallacies of distributed computing are a set of conjectures about distributed computing which can lead to failures in software development.

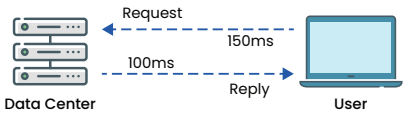
> Nothing is ever completely right aboard a ship.
>
> —William Langewiesche, The Outlaw Sea

8 FALLACIES Of Distributed Systems

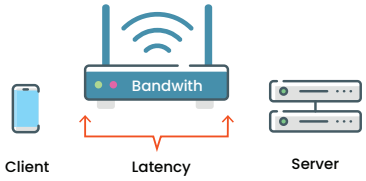
1 The network is reliable



2 Latency is zero

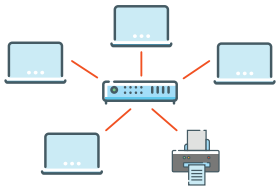


3 Bandwidth is infinite

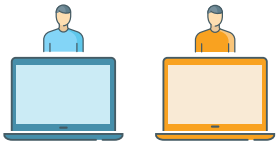


4 The network is secure

5 Topology doesn't change



6 There is one administrator

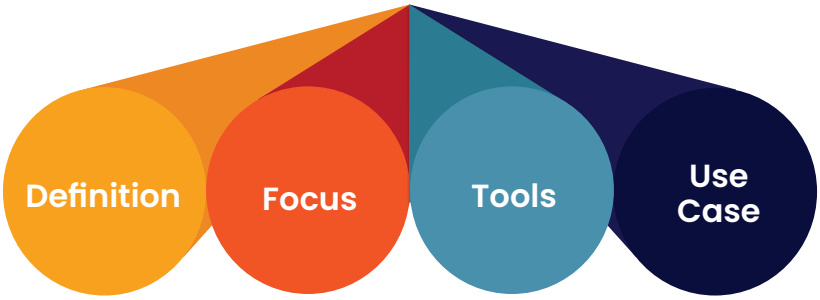


7 Transport cost is zero

8 The network is homogeneous

The ability to observe, visualize, and query your services is a critical tool in determining what problems are happening and in identifying problems before they become incidents that lead to outages.

What is Monitoring?



Definition

Monitoring involves the collection of data and metrics from various components within a system or application to track its health, performance, and availability.

Focus

Observability mainly concentrates on known metrics and predefined thresholds. Monitoring systems typically involve checking for specific events or conditions and generating alerts when these conditions are met or breached.

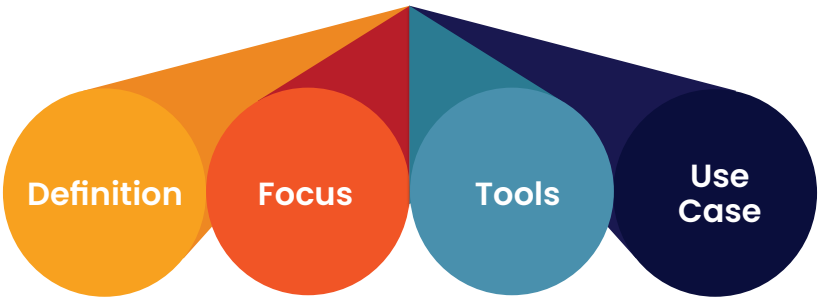
Tools

Monitoring tools provide visibility into the status of systems, networks, applications, and infrastructure. They often rely on metrics, logs, and predefined dashboards to track and display system performance.

Use Case

Monitoring is excellent for tracking predefined metrics and known issues, making it effective for routine maintenance, capacity planning, and identifying known problems.

What is Observability?



Definition

Observability refers to the ability to infer the internal state of a system or application based on its external outputs, allowing for deeper and more comprehensive insights into complex systems. Or in plain English, Can you understand what's happening inside your code and system, simply by asking questions using your tools?

Focus

It goes beyond monitoring by emphasizing the understanding of system behaviors, interactions, and dependencies, especially in distributed and dynamic environments. Observability aims to provide a holistic view of the system's behavior under various conditions.

Tools

Observability tools focus on gathering diverse data sources, including metrics, logs, traces, and other telemetry data. These tools often leverage advanced analytics and visualization techniques to offer a more comprehensive view of system behavior.

Use Case

Observability is particularly useful in complex, dynamic environments, enabling teams to understand the root causes of unknown issues, troubleshoot problems effectively, and gain a deeper understanding of system behavior.

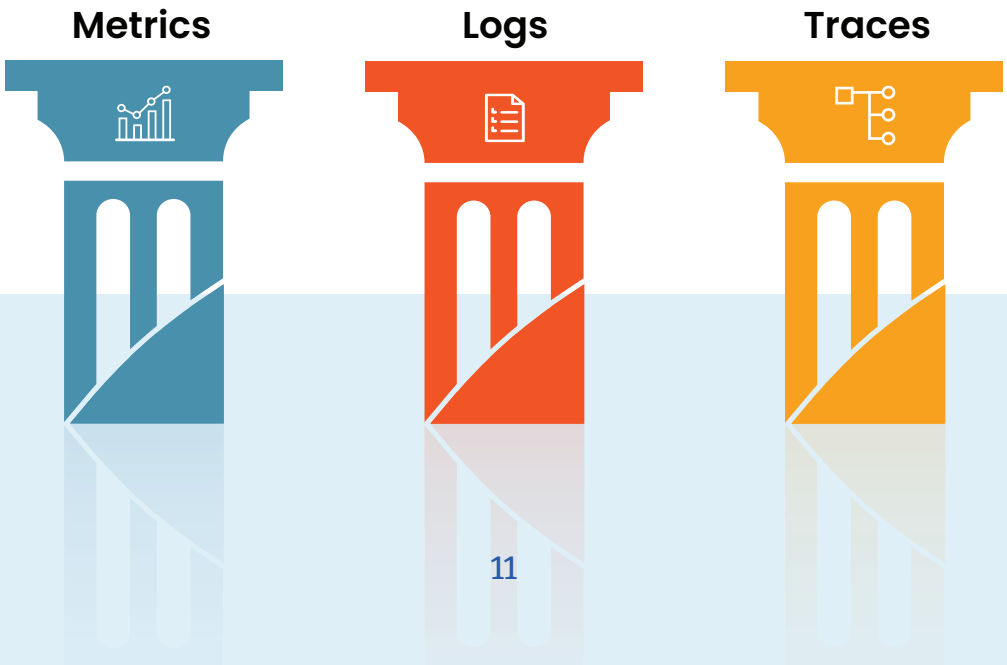
Key Differences between Monitoring and Observability

	Monitoring	Observability
Scope	Monitoring typically deals with predefined metrics and known issues	Observability addresses unknowns and provides a more comprehensive understanding of systems.
Depth of Insight	Monitoring provides specific, predefined data	Observability aims to offer deeper, more contextual insights into complex systems.
Approach	Monitoring is more reactive, triggering alerts based on predefined thresholds	Observability is proactive, aiming to provide a deeper understanding and proactive troubleshooting capability.
Focus	Monitoring is how you manage your known-unknowns, which involves checking values for predefined thresholds, creating actionable alerts and runbooks and so forth.	Observability is how you handle unknown-unknowns, beginning with rich code instrumentation and having well understood systems without shipping new code. Observability needs high cardinality and high dimensionality. (Cardinality refers to the number of unique elements in a set. E.g. Social security numbers have a high cardinality)

Monitoring	Observability
Reactive	Proactive
What happened?	Why did it happen?
Consume information passively	Gain information actively
Fire fighting	Fire prevention
Is my system healthy?	What is my system doing?
Manage your known-unknowns	Handle your unknown-unknowns

Both monitoring and observability are essential in modern IT operations. While monitoring remains crucial for routine maintenance and tracking known issues, observability complements it by providing deeper insights and the ability to troubleshoot complex, unforeseen problems in dynamic systems. Integrating both concepts enables a more comprehensive approach to managing and maintaining systems and applications.

Pillars of Observability



Observability, as a concept in modern system monitoring and management, is often broken down into three primary pillars:

1. Metrics offer a high-level overview of system health over time.
2. Logs provide detailed event-level information.
3. Traces enable the understanding of system interactionsand transaction flows.

1 Metrics: (Do I have a problem?)

Metrics are quantitative measurements or data points that provide information about the performance, health, or behavior of a system over time. These could include CPU usage, memory consumption, request rates, error rates, etc. Metrics help in tracking the overall health and performance of systems, providing a high-level view of system behavior. Metrics are pre-aggregated, numeric measurements and need decision making ahead.

2 Logs: (What is causing the problem?)

Logs are records of events, actions, or observations occurring within a system. These records capture specific details and are often used for troubleshooting, auditing, and understanding system behavior. In observability, logs provide valuable insights into what happened within the system at a particular time, aiding in root cause analysis and debugging. Logs are discrete and not aggregated.

3 Traces: (Where is the problem?)

Traces are detailed records of individual transactions or interactions as they move through a system. They allow for the visualization and understanding of how requests or processes traverse various components and services. Traces help in understanding the end-to end flow of transactions, identifying bottlenecks, and optimizing system performance. Traces represent the flow of a request (request-scoped). Traces depict the causal chain of events between different components.

By leveraging these pillars collectively, organizations gain a comprehensive understanding of their systems' behavior, which aids in diagnosing issues, optimizing performance, and maintaining the reliability and efficiency of complex distributed systems.

Choosing Useful Metrics

There is a plethora of metrics available that may be overwhelming. Don't become data rich and information poor. Averages can be misleading. Trying to average averages is a well-known statistical trap. Don't create alert fatigue. Checking disk space for free space every minute is not necessary.

Google's SRE book details the 4 golden signals that you simply need to track if you want your system to be the best it can be as it scales.

The Four Golden Signals

Latency
It means the time that passes between the moment you request a service and the moment you actually get it.

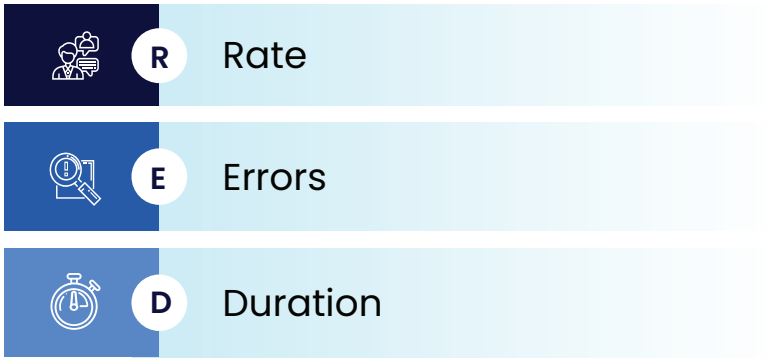
Traffic
It refers to how much activity is present in the application.

Errors
Meaning the rate at which your requests are failing.

Saturation
It measures how full your service is.

The RED and USE patterns are useful frameworks used in monitoring and observability to categorize and define key metrics for understanding system performance.

RED Method



Rate

The Rate metric focuses on measuring the rate of requests or transactions processed by a system within a given time frame. It helps in understanding the workload or demand on the system. Since the total number of requests only ever goes up, it's more useful to look at request rate: the number of requests per second, for example.

Errors

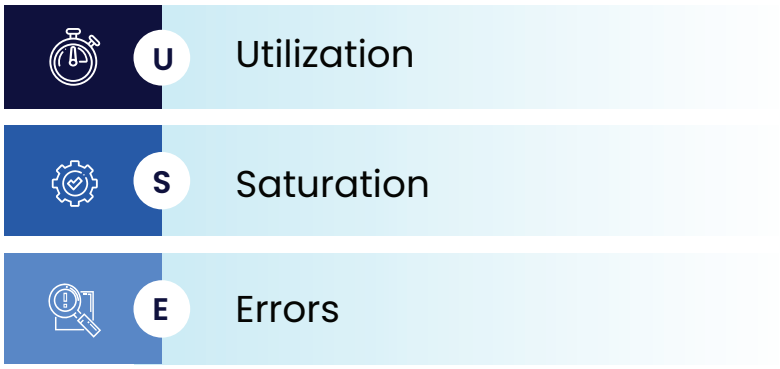
Errors represent the rate of failed or erroneous requests encountered by the system. This metric provides insights into the number of requests that result in errors, helping in identifying and troubleshooting issues.

Duration

Duration refers to the time taken for a system to process requests or transactions. It measures the latency or response time and helps in understanding the performance of the system from the users' perspective.

The RED method emphasizes tracking these three key metrics—Rate, Errors, and Duration—to gain insights into the performance, reliability, and efficiency of systems. It's particularly useful for identifying performance bottlenecks and understanding system behavior under varying workloads. It is more related towards services.

USE Method



Utilization

Utilization measures the percentage of system resources (CPU, memory, disk, network) being used at any given time. It indicates how much of the available capacity is being utilized and helps in capacity planning.

Saturation

Saturation refers to the degree of workload a system is experiencing concerning its maximum capacity. It measures the queue lengths, backlogs, or contention for resources, indicating when systems are overloaded.

Errors

Similar to the 'Errors' in the RED method, this metric tracks the rate of errors or failures occurring within the system.

The USE method focuses on understanding system performance by monitoring Utilization, Saturation, and Errors. It helps in identifying resource bottlenecks, determining system limits, and optimizing resource allocation. It is more related to hardware/infrastructure.

Both RED and USE patterns offer valuable frameworks for selecting and categorizing metrics that are essential for gaining insights into system behavior, performance, and reliability. They provide a structured approach to monitoring that enables efficient troubleshooting, capacity planning, and optimization of systems and applications.

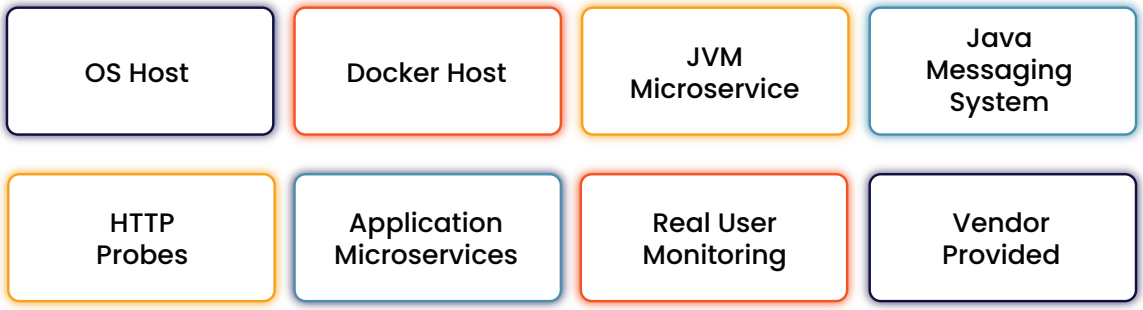
Observability Case Study

The Customer is a leading provider of end-to-end HR and Payroll solutions with a global reach. The software architecture is built for multitenancy based on microservice architecture. It is hosted on AWS with self-hosted Kubernetes as the orchestration platform. Webserver backend is in Java while data is stored in PostgresDB.

The Customer was facing challenges in daily operations – performance of the application was poor and there were frequent outages leading to end-user dissatisfaction.

DevOn helped design, architect and implement the Observability Stack.

We started monitoring the below components in the stack :



OS Host

Hardware node metrics like RAM, CPU, Network traffic etc

Docker Host

Container specific resources RAM, CPU, etc.

JVM Microservice

Since microservices are Java apps, we monitored JVM that executes the base code. We have metrics like garbage collector usage, thread count, heap and non heap usage etc.

Java Messaging System

Since we use JMS system to ensure communication between microservices, we monitor the broker metrics like total number of messages enqueued, total number of messages dequeued, number of messages by queue/ topic etc.

HTTP Probes

Every microservice exposes a healthcheck endpoint. This endpoint needs to be monitored in order to ensure the availability of the app

Application microservices

Application functional metrics like ongoing payroll, etc

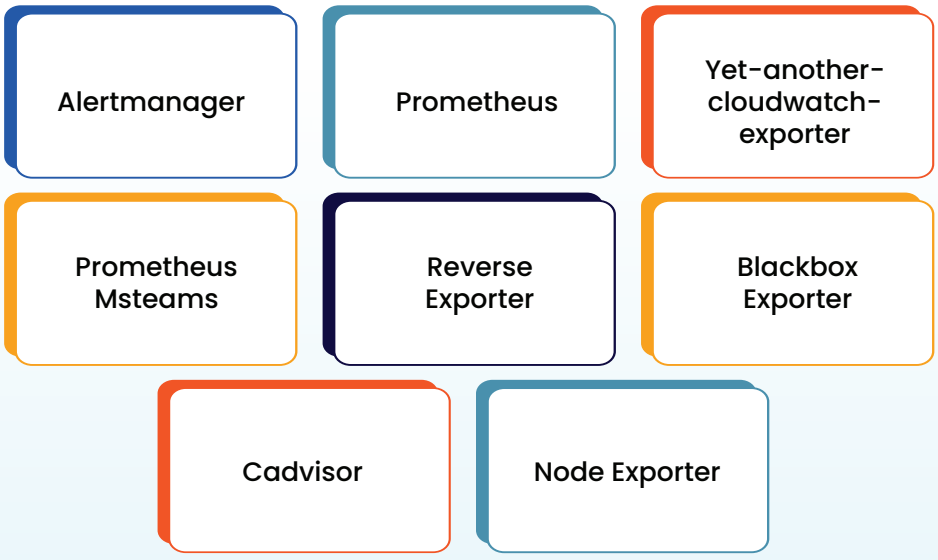
Real User Monitoring

We monitor real user interactions with the app, it collects and saves all actions done by a user and transform it into useful metrics

Vendor provided

Our cloud hosting platform is AWS and we monitor Cloudwatch metrics for RDS and Elasticsearch.

We implemented the below services in our cluster to collect metrics :



Alertmanager

Alertmanager handles alerts sent by client applications such as the Prometheus server. It takes care of deduplicating, grouping, and routing them to the correct receiver integration such as email

Prometheus

Prometheus is a systems and service monitoring system. It collects metrics from configured targets at given intervals, evaluates rule expressions, displays the results, and can trigger alerts when specified conditions are observed.

Yet-another-cloudwatch-exporter

YACE exports Cloudwatch rds metrics to a prometheus metrics formats. It takes advantages from grouped api calls to optimize costs

Prometheus-Msteams

A lightweight Go Web Server that receives POST alert messages from Prometheus Alert Manager and sends it to a Microsoft Teams Channel using an incoming webhook

Reverse Exporter

The reverse_exporter logically decodes its target exporters on each scrape, allowing them to be presented as unique metrics to Prometheus.

Blackbox Exporter

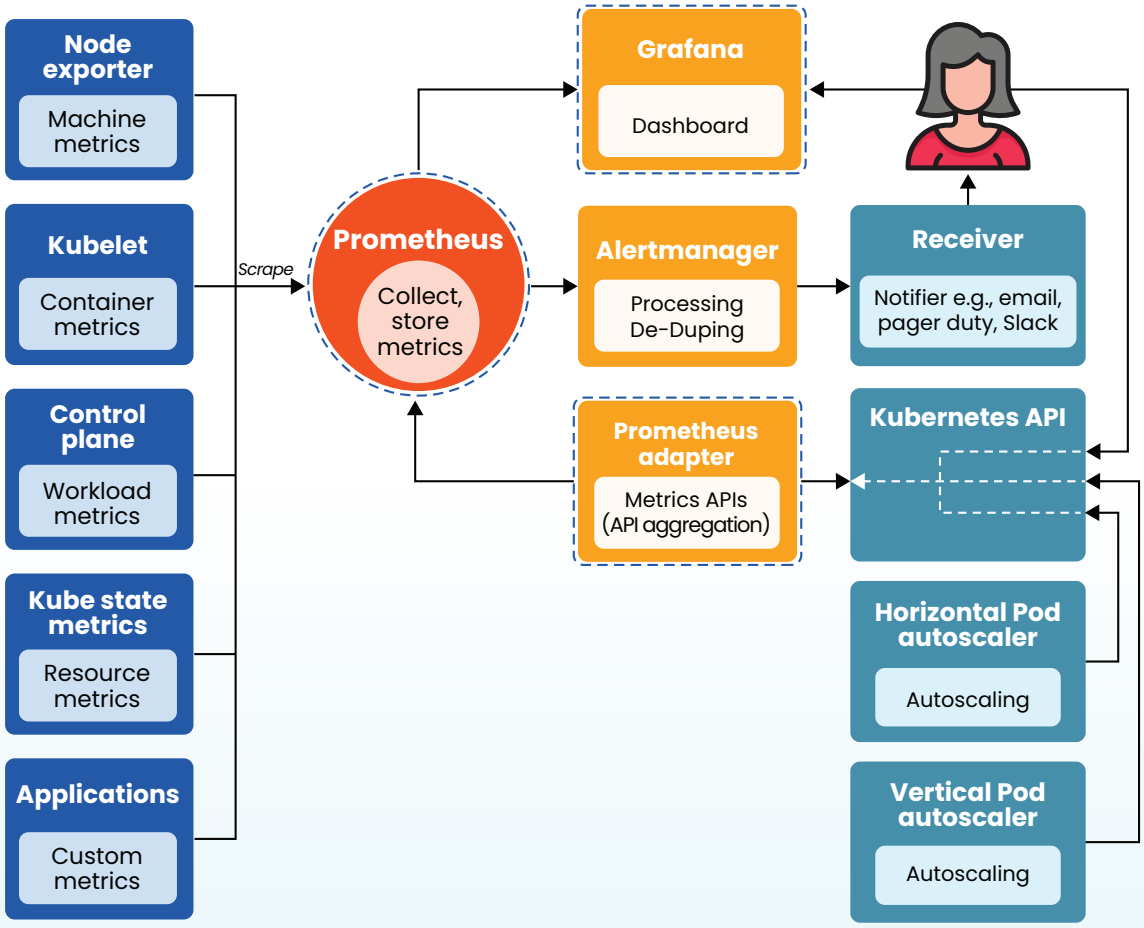
The blackbox exporter allows blackbox probing of endpoints over HTTP, HTTPS, DNS, TCP and ICMP.

Cadvisor

cAdvisor (Container Advisor) provides container users an understanding of the resource usage and performance characteristics of their running containers. It is a running daemon that collects, aggregates, processes, and exports information about running containers. Specifically, for each container it keeps resource isolation parameters, historical resource usage, histograms of complete historical resource usage and network statistics. This data is exported by container and machine-wide

Node Exporter

Prometheus exporter for hardware and OS metrics exposed by *NIX kernels, written in Go with pluggable metric collectors.



We use the tools below for Monitoring and Observability

Grafana

Grafana is the open source analytics & monitoring solution. Grafana allows you to query and visualize on and understand your metrics no matter where they are stored. Create, explore, and share dashboards with your team and foster a data driven culture.

Datadog

Datadog is a cloud-based monitoring and analytics platform that provides organizations with tools for monitoring the performance of their applications, infrastructure, and networks.

Karma

Karma is a UI for Alertmanager, useful for browsing alerts based on labels and managing silences. It can also aggregate alerts from multiple Alertmanager instances.

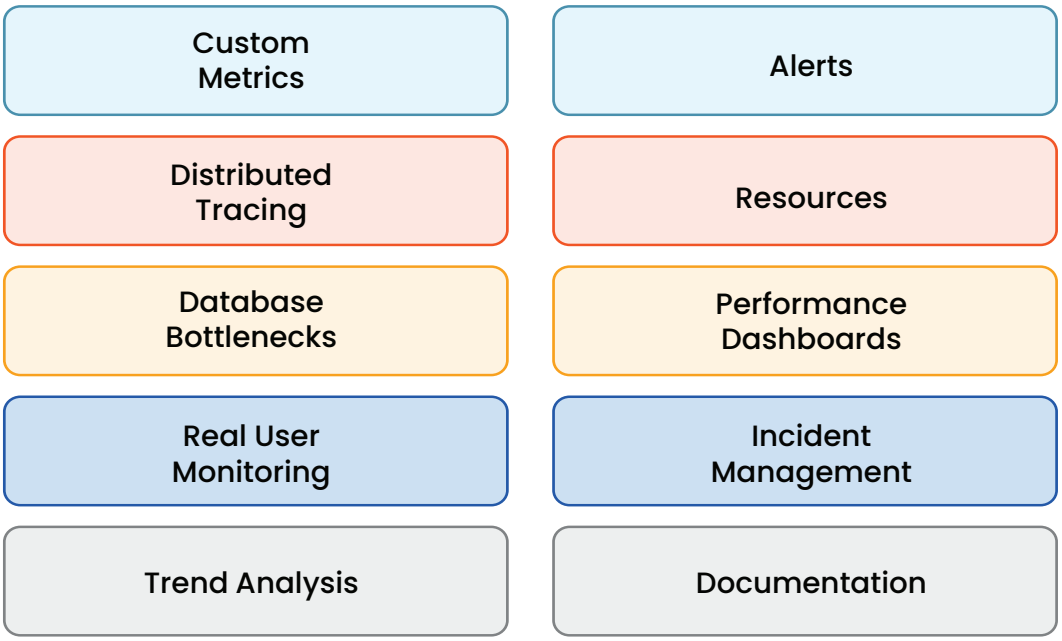
Graylog

Graylog is an open-source log management and analysis platform that helps organizations collect, index, and analyze log data from various sources in real-time.

Cloudwatch

Amazon CloudWatch is a monitoring and observability service provided by Amazon Web Services (AWS). It is designed to help users collect and track metrics, collect and monitor log files, and set alarms based on specified thresholds.

Best Practices



Custom Metrics

We understood that out-of-the-box metrics may not be sufficient for our service objectives. We created custom metrics and used them in our hpa (horizontal pod autoscaling).

Distributed Tracing

We started analyzing traces using Distributed Tracing in Datadog to find bottlenecks in the performance of our services.

Database Bottlenecks

We deleted unused indexes. Optimized the auto-vacuum.

Real User Monitoring

We started tracking user behavior using Real User Monitoring (RUM) feature in Datadog. E.g. we found user tended to double click leading to anomalies.

Trend Analysis

We started performing trend analysis – for e.g. we found an automated test failing daily at a specific time – we found the culprit to be a pg-ldap sync process that started at the same time.

Alerts

We revisited our alerts and kept adding new alerts. E.g. We were monitoring memory of the pod, but not the JVM heap memory – these were added.

Resources

We revisited the request and limits set in the pod manifests. As the platform grew, our older thresholds needed to be updated

Performance Dashboards

We explored tools like pghero which is a performance dashboard for postgres. This helped identify connections, slow indexes, duplicate indexes, and slow executing queries.

Incident Management

We started an Incident Management process – daily rosters to monitor alerts were set in place.

Documentation

We documented our Observability stack and details of alerts. Any change made to the cluster was documented for traceability.

With the above Observability stack in place, we were able to start making improvements towards enhanced stability and reliability of services.

“

With the above Observability stack in place, we were able to start making improvements towards enhanced stability and reliability of services.

- Customer Head of DevOps

Key Results

- In the first 3 months itself of implementing the Observability platform, the mean time to resolution (MTTR) improved by 15%. Even more significantly, the number of issues reported decreased by 33% thanks to features such as proactive alerting and anomaly detection.
- Reduced eight separate monitoring tools to a single unified platform
- Quick adoption across Dev and Ops teams
- Cost-effectively ingest and index 177M log events of logs each day

DevOn has thus positioned itself to be a trusted partner for organizations to accelerate their journey to high performance with full stack observability resulting in fewer outages, faster MTTD, faster MTTR, lower outage costs and higher ROI

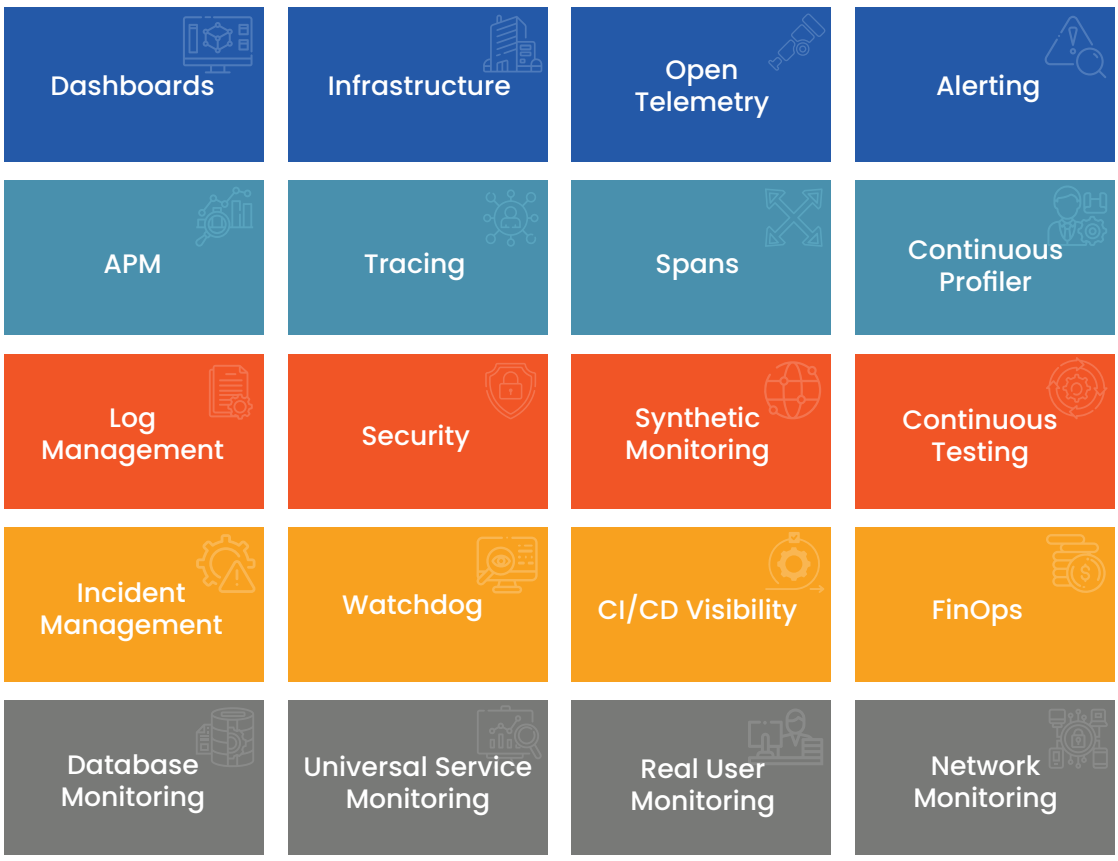
From Splunk State of Observability 2023 Global Research Report, respondents report that observability solutions are having a positive impact across the board.

Observability at Enterprise Scale

Observability at Enterprise scale takes basic observability to the next level. It allows us to proactively track and optimize our entire IT stack's performance, from the backend infrastructure to the application logs and traces. It lets us unify and visualize all our performance data in real time using a single pane of glass, enhancing collaboration across teams and troubleshooting issues more effectively.

An Observability Platform at Enterprise scale can correlate interactions between different systems and applications, ingest data from multiple sources, auto instrument and avoid patchwork quilt of legacy tools.

Here are some key aspects and strategies for achieving observability at enterprise scale:



Dashboards

Visually track, analyze and display key performance metrics. Typical dashboards are auto suggested. Support for mobile devices and also TV mode.

Infrastructure

Track all your host instances, containers and processes centrally

OpenTelemetry

OpenTelemetry is an open-source observability framework that provides a set of APIs, libraries, agents, and instrumentation to instrument, generate, collect, and export telemetry data from applications and services. It aims to standardize and simplify the instrumentation and collection of telemetry data in distributed systems.

Alerting

Configure monitors, notify your teams, and manage alerts at a glance

APM

Application Performance Monitoring (APM) provides many tools to troubleshoot application performance and correlate it throughout the product, enabling you to find and resolve issues in distributed systems.

Tracing

Collection of operations that represents a unique transaction handled by an application and its associated services. A trace tracks the amount of time spent processing a request. A trace consists of one or more spans.

Spans

Represents a single unit of work within a distributed system. It is the primary building block of a distributed trace.

Continuous Profiler

A profiler shows how much memory and cpu each function is utilizing by collecting data about the program as it is running. This is great for debugging performance problems. Use the profiling results to optimize your code.

Log Management

Collecting logs from multiple sources, live tail facility, generate metrics and alerts from logs.

Security

Provides real-time threat detection, and often provide typical features of a SIEM (Security Information and Event Management) system

Synthetic Monitoring

Allows you to observe how your application is performing using simulated requests and actions from different locations around the world. These can be multi-step. Audit uptime, flag issues restricted to specific location and test application performance.

Continuous Testing

Helps you analyze test failures and identify flaky tests. You can instrument your test code too and generate traces from your testing frameworks as they are getting executed.

Incident Management

Tracking and communicating about an issue you've identified with your metrics, traces, or logs. Declare incidents in response to issues all without switching context.

Watchdog

Automating preliminary investigations during incident triage. Correlate between interdependencies for performance anomalies and draw causal relationships between symptoms often using AI and ML.

CI/CD Visibility

Provides information about CI (plan, code, build, test) and CD (deploy, release) pipelines results in addition to data about performance, trends, and reliability. Enables teams to quickly triage build failures, pipeline issues, monitor trends in test execution times, and track differences between releases.

FinOps

Provides insights for teams to see how changes to infrastructure can affect costs. It enables you to take ownership for your cloud usage. Trends and variation analysis can help to understand any increase in costs

Database Monitoring

Provides information about connected databases. Dig into query performance metrics, find slow running queries, to understand the health and performance of your databases and troubleshoot issues as they arise.

Universal Service Monitoring

Provides visibility into your service health metrics universally across your entire stack without having to instrument your code.

Real User Monitoring

RUM (Real User Monitoring) observability refers to the practice of monitoring and analyzing user interactions with web applications or services in real-time. It also supports Session Replay where you can visually replay the web browsing experience of the user. Monitor API response times, details of what browsers, OS and devices are being used.

Network Monitoring

Provides visibility into on-prem and virtual network devices, such as routers, switches, and firewalls.

References

» Guide - Achieving Observability | Honeycomb
» Slowdown is the new outage (SITNO) - Application Performance Monitoring Blog | AppDynamics
» tag-observability/whitepaper.md at main · cncf/tag-observability · GitHub
» Google - Site Reliability Engineering (sre.google)
» Whitepaper: Continuous Application Modernization - DevOn
» Book: Cloud Native DevOps with Kubernetes (O'Reilly)
» Splunk: The State of Observability 2023

Takeaways

Observability is a rapidly evolving field. The ability to observe any software system is critical. No application should be considered production-ready without observability in place. Increased focus on Security and AIOps is also driving the need for Observability. Organizations want tool consolidation, unified telemetry data and prefer a single unified platform. There are clear business benefits of Observability - Full stack observability results in fewer outages, faster MTTD, faster MTTR, lower outage costs and higher ROI.

"The trick is to gather the right data in the first place, process it in the right way, use it to answer the right questions, visualize it in the right way, and use it to alert the right people at the right time about the right things." (From Book: Cloud Native DevOps with Kubernetes)

AUTHOR

Prashant Sharma
Principal Consultant
DevOn

About DevOn

DevOn is a technology company focused on helping organizations achieve high performance in their remote software delivery teams. We offer innovative solutions designed to enhance team efficiency and effectiveness, regardless of location. Our seven key principles of effective communication, strong leadership, clear expectations, regular feedback, continuous improvement, recognition and reward, and results-focus serve as the foundation for building successful remote teams. Our team of experienced professionals utilizes cutting-edge technology to ensure seamless collaboration. With DevOn, organizations can improve the performance of their remote software delivery teams and reach their goals faster.

Awards & Recognition

3 Times Great Place to Work Certified

Certification based on **Trust Index 94%** – a comprehensive employee survey and culture audit

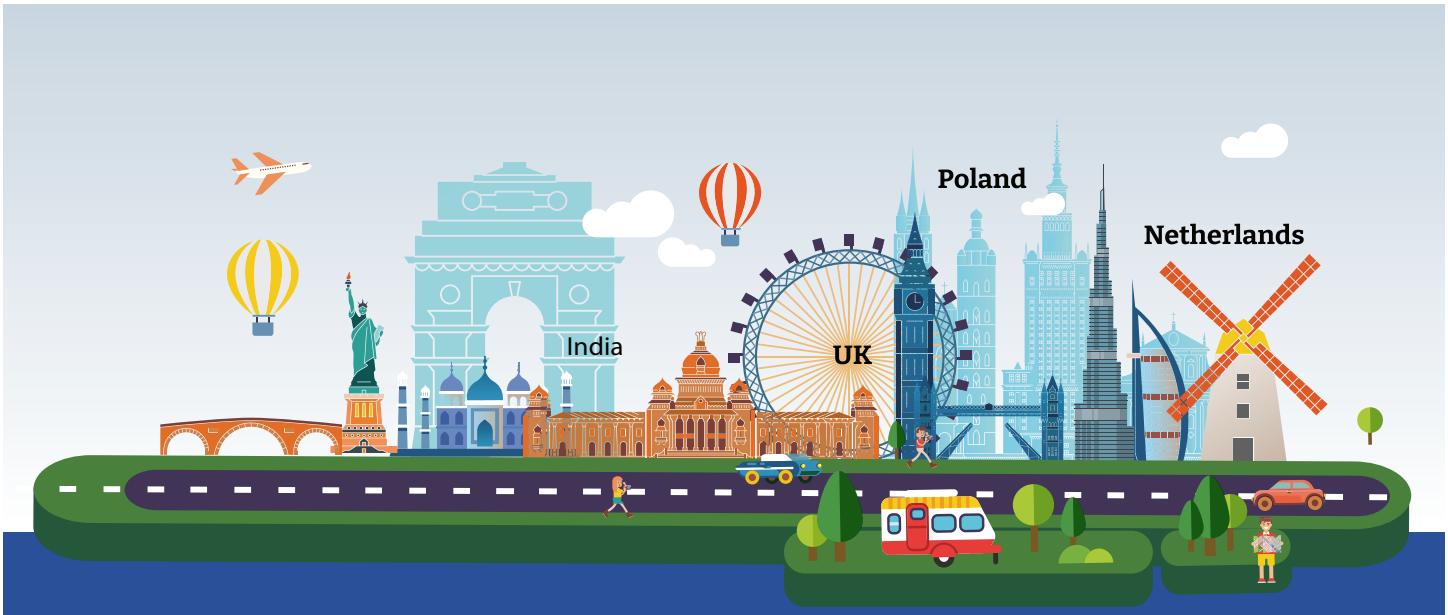
Top 10 Inspiring Workplaces 2023

Ranked No.4

Top 50 Mid-size Workplaces in India

National Best Employer Brands Award 2022

India's Best Workplaces for Millennials™ 2023



CONTACT US

Speak with one of our experts

Our insights can help you take advantage of change. If you're looking for fresh ideas to address your challenges, please feel free to reach out to us for a quick brainstorm.

THE NETHERLANDS

Brassersplein 1, 2612 CT Delft

015 241 1900
info@devon.nl
http://www.devon.nl

INDIA: BANGALORE

2A-West Tower, Embassy Tech Village, Outer Ring Road, Deverabeesanahalli Village, Varthur Hobli, Bellandur, Bengaluru, Karnataka 560087

+91 80672 98000
https://devon.global

INDIA: GURGAON

6th Floor, MM Towers, Plot No. 8 & 9, Phase IV, Udyog Vihar, Sector 18, Gurugram, Haryana 122001

+91 6462 203 377

UNITED KINGDOM

7 Three Rivers Business Park, Felixstowe Road, IP10 0BF, Foxhall, Ipswich, United Kingdom

+44 20 3318 2856

POLAND

Spaces Fabryka Kart 1,2,3,4,5 piętro, 13 Cieszynska street Krakow, 30-015 Poland

+48 733 186 001

To be a trusted partner for professionals and organizations to accelerate their journey to high performance