StackState

Dynamic Observability for Cloud Native Applications and Environments

By Bernd Harzog

Table of contents

Dynamic Observability for Cloud Native Applications and Environments

The rate of change in applications, the complexity and diversity of the infrastructure supporting these applications, and the dynamic behavior in modern cloud-based infrastructures create unprecedented challenges for Site Reliability Engineers (SREs) seeking to ensure the performance and reliability of their online services.

The Challenges Created by Digital Transformation

Enterprises globally are implementing their core customer, prospect, partner, supplier, and employee facing processes in software. This means that every enterprise is becoming a software company with the need to build software, test software, operate software in production, and most importantly continuously enhance this software to keep it competitive and to respond to requirements from the users of the software.

This demand on the part of companies worldwide for software engineering well exceeds the supply of available software engineers and results in continuous stream of innovations designed to speed the development of software and the delivery of software into production:

- *Agile Development breaks development into smaller chunks (sprints) allowing for continuous incremental progress in software development.*
- *DevOps makes supporting software in production more effective and efficient*
- *CI/CD automates much of the process of delivering software into production*
- *New languages like Python, PHP, Node-JS, and Go were invented to speed the development of the classes of applications for which these languages are appropriate*
- *New runtimes like Spring Boot, OpenShift and Pivotal take progressively more of the burden of building and running infrastructure software off of the shoulders of the developers*
- *Docker allows applications to be isolated in containers along with their supporting libraries which smooths the process of testing and integrating new releases into production*
- *Kubernetes automates the process of orchestrating groups of containers as needed to respond to changes in demand and other conditions*
- *Databases proliferated in order to meet the needs for various data models, scalability needs, performance needs, data types and redundancy requirements*

In summary the response to Digital Transformation has created the following unprecedented dynamics:

- *A very high pace of innovation in processes, tools, and platforms*
- *A high rate of updates of application software in production*
- *A very diverse set of languages*
- *A very diverse set of runtimes, supporting software and databases*
- *A dynamic execution environment in which services are rapidly scaled up and down or moved from one computing resource to another*

These dynamics together make it extremely challenging to support scaled out and highly dynamic applications and to ensure a consistent user experience.

The Consequences of Poor Digital Experience

All of this software must be reliable and provide excellent end-user experiences or else the enterprise suffers the following adverse consequences:

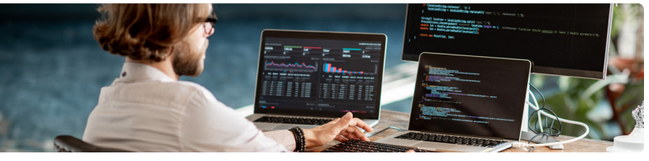
- ***Reductions in online revenue*** *– 63% of consumers say they often abandon a brand when the online experience is poor*
- ***Impacts to online company reputation*** *– 72% of consumers say they are loyal – until they have a bad experience*
- ***Impacts to customer experience*** *– 54% of U.S. consumers say customer experience at most companies needs improvement*

Furthermore, internally time is wasted addressing issues instead of enhancing online functionality and business performance. Each month, companies experience 5 or more outages, lasting an average of 200 minutes and involve 25 or more employees to resolve. Therefore the time and people cost of responding to incidents is far higher than it should be.

The Current State of Monitoring and Incident Management

In order to combat the negative impacts of poor digital experiences, modern enterprises have turned to monitoring tools to help identify and prevent performance and reliability issues. In other words, if you want it to work, you have to monitor it. You have to monitor each layer or component in your stack to ensure the following:

- *That it is working (up and available to do work)*
- *That it is performing the work required in the required interval of time (latency or response time)*
- *That it is performing the amount of work required per interval of time (throughput)*
- *That it is not experiencing errors while performing the required work*
- *That it is not experiencing contention for resources (CPU, memory, network I/O and storage I/O) with other software running in the environment*
- *That the amount of resources available are sufficient to meet the needs of the component (capacity)*



There are two basic approaches to monitoring business critical applications and services in production. The first is to monitor the infrastructure for the applications. In the world before virtualization and the cloud, this means monitoring the physical severs, network devices, and storage devices that comprised the IT environment. In the modern virtualized and cloud world, infrastructure means all of the software running in the virtualization or cloud environment (the operating systems, application frameworks, databases, containers and orchestration s ystems) that supports the applications.

The second approach is to monitor the operation of the applications themselves. This typically involves injecting an agent into the application or the run time of the application and then measuring the performance, throughput and error rate of the actual transactions executed by the users of the application. APM (Application Performance Management) tools are widely used to monitor applications in production, as are a variety of open source alternatives to commercial APM tools like OpenTelemetry and Prometheus.

Each of these tools feeds its alerts and incidents into some form of an event management, incident management, or alert notification system.

This existing monitoring and incident management process leads to the following issues for enterprises:

- *Every enterprise owns multiple monitoring tools each monitoring a layer or a silo in the environment*
- *The only integration of these monitoring tools is typically that their incidents or alarms are forwarded to a centralized incident management system or incident response system*
- *Incidents do not contain the information needed to tie incidents to each other, requiring extensive drill down into various tools often occurring in physical or virtual war room meetings.*
- *The number of incidents and the time it takes to solve them is therefore generally unacceptable to the business.*
- *Therefore the cost in people time and the cost in terms of business impact of incidents are both unacceptable to owners of critical online business services.*

Enter Dynamic Cloud Observability

The traditional focus of Observability is on metrics, logs, and traces. Countless vendors who support some combination of metrics, logs, and traces all now claim to be Observability vendors. However, most of these vendors supported metrics, logs, and traces before the term Observability became fashionable, so for these vendors Observability is just a new way of talking about existing capabilities.

Dynamic Cloud Observability breaks new ground by adding the following unique capabilities to an Observability platform or solution:

- **Real-Time Build (Update) Tracking** – Tracking of each new update in production, and its resulting changes in platform configuration and relationships.
- **Real-Time Configuration State Tracking** – Tracking of the configuration of the application and its entire supporting virtualization and cloud infrastructure in real-time over time.
- **Real-Time Relationship (topology) Change Tracking** – Tracking of the dependencies between transactions, microservices, applications, containers, Kubernetes Pods, Kubernetes Nodes, virtualization and cloud platforms, and all of the resources (compute, memory, networking, and storage) that support each application in real-time over time. This means knowing exactly what these relationships are and how they changed before, during and after each incident.
- **Root cause based upon AI AND Related Changes.** Dynamic Cloud Observability means that the AI knows with certainty that a set of objects are related to each other and that they all support and affect the transaction of interest. This avoids the false alarms (false positives) that result from the AI confusing correlation with causation.
- **Incident Prevention** – Dynamic Observability uses early detection of anomalies to allow for incidents to be entirely prevented and avoided.

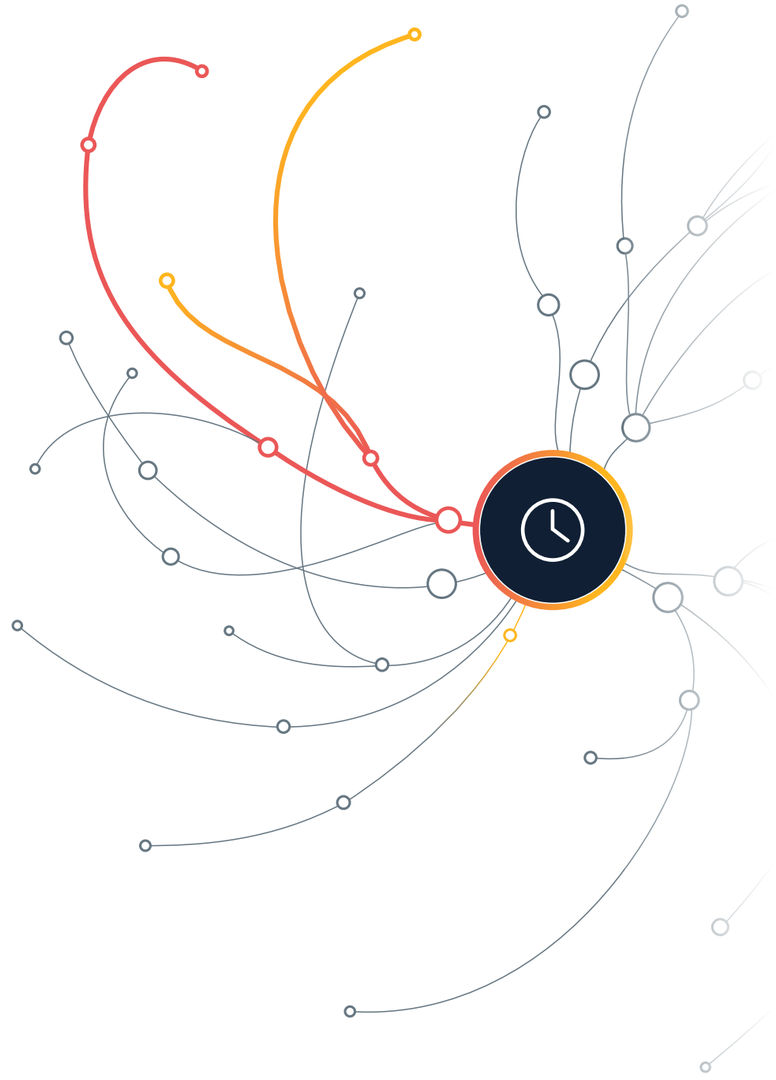
The Value of Dynamic Observability

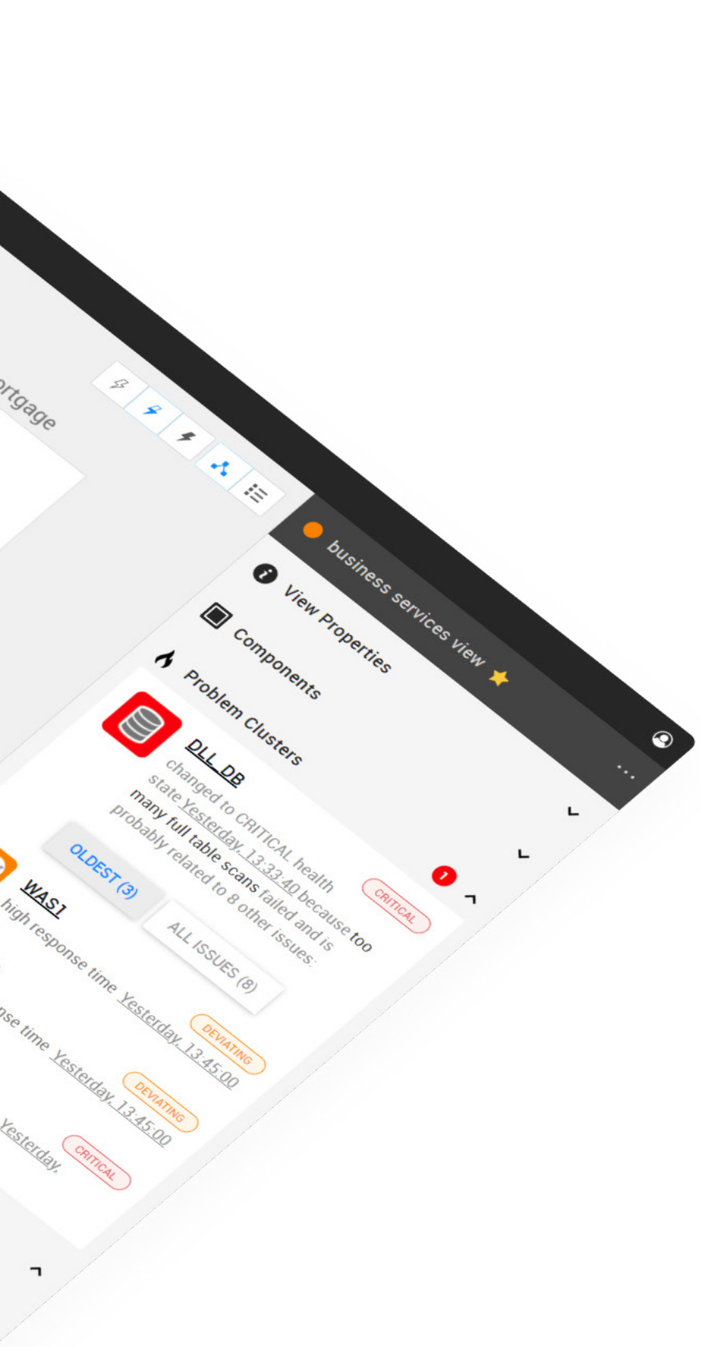
Dynamic Observability automatically adds the information needed to resolve issues into the existing incident management system. This dramatically improves the existing incident management process without disrupting it by replacing either the underlying monitoring tools, or the incident management tools themselves. These improvements in the incident management process lead to:

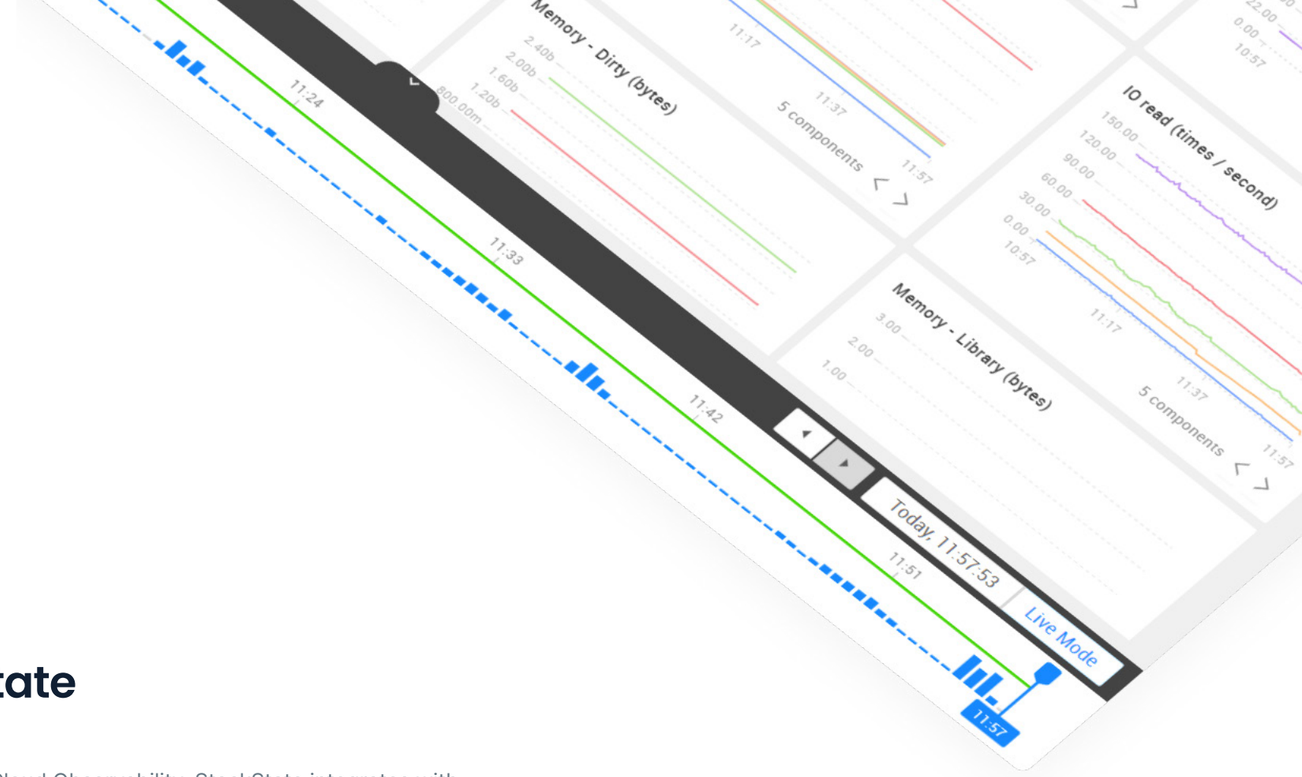
- **60% reduction in the time to address incidents**
- **20% fewer staff needed to fix incidents**
- **65% decrease in the number of incidents per month**
- **And 30% reduction in the cost per incident**

All of which leads to dramatic improvements in online revenue, online reputation, and online customer experience.

The deterministic nature of the relationships and configuration changes over time are also the necessary pre-conditions to being able to automate problem resolution as it is essential to know what impacted what and what changed in order to accurately take automated actions.







About StackState

StackState delivers Dynamic Cloud Observability. StackState integrates with cloud platforms, Kubernetes, Prometheus and incident management systems to add the certainty and richness of relationships, configuration changes and AI-based diagnostics to the existing incident management process. This uniquely leads to Deterministic Root Cause, which helps IT teams prevent and solve problems more quickly and efficiently.