



AHEAD

How Observability & SRE Support Platform Engineering



splunk>
a cisco company

Platform engineering is crucial in today's software development. It helps streamline app deployment, management, and scaling via a shared platform, making it easier for developers to innovate by reducing infrastructure complexity. However, cloud computing and microservices have helped to create significant challenges in the management of complex, distributed systems, such as ensuring scalability, security, and high availability. The two primary factors in dealing with these obstacles are **Observability** and **Site Reliability Engineering (SRE)**.

Observability provides a comprehensive view of system behavior and application health by analyzing metrics, logs, and traces – this enables proactive issue resolution and performance optimization. On the other hand, SRE focuses on creating reliable, scalable systems. SRE practices, such as defining Service Level Objectives (SLOs), help balance innovation with system stability.

In this paper, we'll explore the distinct roles that observability and SRE play with regard to software development and demonstrate the ways that they work in tandem to inform best practices for innovative and efficient platform engineering.

By 2024, it is anticipated that businesses will use observability tools **30% more frequently**, demonstrating the growing awareness of their vital role in assisting users across complex digital environments.

Understanding Observability as an Essential Part of Platform Engineering

Platform engineering focuses on creating and managing platforms that enable developers to build and deploy applications efficiently. These platforms offer a standardized toolkit, simplifying and enhancing development by providing necessary services and tools.

To ensure a robust and agile development environment, platform engineering rests on six key pillars:



SECURITY



PIPELINE (VCS, CI/CD)



PROVISIONING



CONNECTIVITY



ORCHESTRATION



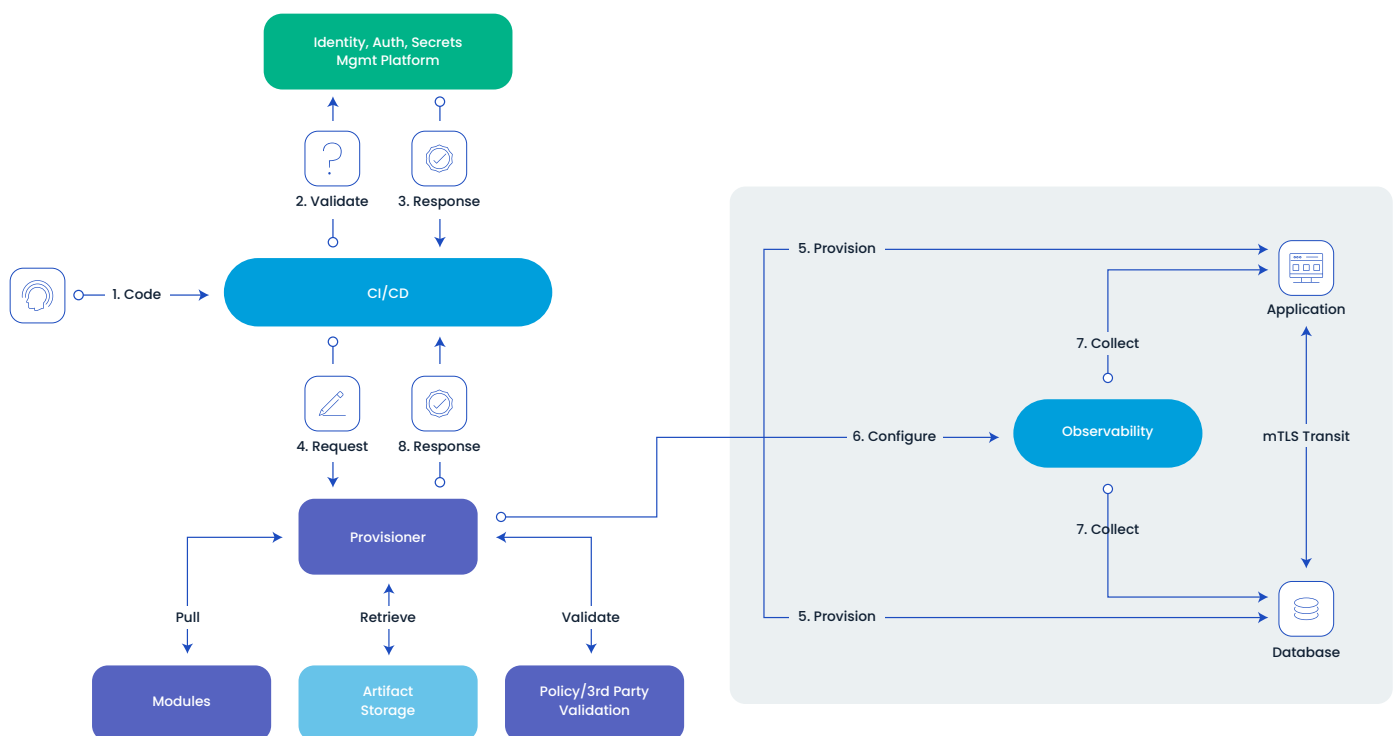
OBSERVABILITY

Among these pillars, observability is the foundation. Without deep visibility into the platform's inner workings, it's impossible to guarantee an application's reliability, performance, and overall health – all of which developers depend on. Observability refers to the understanding of how a system functions through observation of its outputs. In platform engineering, this involves monitoring and analyzing the performance of the platform itself and the applications it hosts. Observability is crucial in platform engineering, as it supports DevOps teams in achieving key goals, such as improving release quality, increasing release frequency, and enhancing operational efficiency.

According to [DEJ's 2022 report](#), organizations are losing almost \$7.2 million a year as a result of lacking observability.

Establishing observability early in every environment is essential, as it prevents the buildup of different solutions and siloed data and facilitates collaborative problem-solving. Platform engineers must prioritize solutions that minimize cognitive load and tool sprawl – doing so ensures documentation, consistency, and ease of deployment across environments.

Now, with a better understanding of observability's role platform engineering, let's take a look at the eight key steps of an enterprise-level observability workflow:



[Source](#)



01. **Code Commit:** A developer makes a code change and commits it to the repository. Depending on their roles and permissions, they might have direct access to modify the network control plane.
02. **Validate:** The Continuous Integration/Continuous Deployment (CI/CD) system checks the commit by requesting authentication and authorization (AuthN and AuthZ) from the Identity Provider.
03. **Identity Provider Response:** If the Identity Provider confirms the developer's credentials and permissions, the CI/CD pipeline initiates predefined tasks like testing, building, and deploying the code.
04. **Request:** The system that provisions resources (the provisioner) begins its process. This includes building necessary modules, fetching artifacts, and validating policies against internal and external policy engines to set up the requested resources.
05. **Provision:** The necessary infrastructure for the application or service is provisioned, set up, and ready for use (if it wasn't already available).
06. **Configure:** After provisioning, the provisioner configures observability resources. This step ensures the infrastructure is available, monitored, and observable.
07. **Collect:** With the observability resources configured, the system collects metrics and trace data. This collection is based on the emitters and aggregators set up in the previous step, providing insight into the system's performance and health.
08. **Response:** The final step involves the provisioner completing its tasks and reporting back to the CI/CD system. This completion might trigger further actions, like security scanning or integration testing, to ensure everything works as expected.

This workflow outlines a systematic approach to integrating observability into the software development and deployment process, ensuring that systems are functional, but also measurable and maintainable.

REAL-WORLD EXAMPLE:

Tesla and Usage-Based Insurance

Tesla serves as a compelling example of how applied observability can be leveraged to create innovative business models. Tesla cars have many sensors and use Autopilot to watch how you drive. Every month, they give each driver a “Safety Score.” This score shows how safely you’re driving.

Tesla doesn’t rely on traditional factors like age or location to determine insurance premiums. Instead, by leveraging the “observable” real-time driving data, they can offer personalized insurance rates based on actual driving habits. Safer drivers, as measured by the Safety Score, can potentially receive significant discounts on their premiums (20% to 60% according to Tesla).

Benefits of Observability in Platform Engineering

Observability offers several benefits to platform engineering. According to Enterprise Strategy Group, advanced observability deployments significantly reduce downtime costs by 90%, resulting in annual costs of **\$2.5M compared to \$23.8 million** for observability beginners.

Let’s drill down further on some of the key advantages seen by enterprises who prioritize observability:

Proactive Issue Detection

Observability equips platform engineering teams with tools and methodologies to actively monitor and analyze system data in real time. By understanding standard operational patterns, teams can detect anomalies or deviations that may indicate potential issues or system degradations.

Better Use of Automation with Machine Learning

Observability data can be analyzed to detect patterns and automate system repairs, saving significant time and resources.

Enhanced End-User Experience

In-depth performance monitoring enables the organization to optimize systems for better experiences that drive user retention and satisfaction.

Faster Troubleshooting & Root Cause Analysis

Observability tools collect and analyze lots of data to clearly show how systems are working. This makes it easier to find and understand problems quickly. As a result, fixing issues takes less time, improving system reliability and efficiency.

Improved Performance Optimization

Observability offers detailed insights into the performance of different system components. By continuously monitoring metrics, logs, and traces, teams can identify performance bottlenecks, inefficient resource usage, and opportunities for optimization.

Observability Criteria for Platform Engineers

For platform engineers, an ideal observability solution should be:

- **Fast and Easy to Implement:** The observability solution should be straightforward to deploy across various teams and environments. A standardized process ensures quick adoption and minimizes setup time, which is essential for maintaining agility.
- **Easily Customizable:** Flexibility is vital. The solution must adapt to different teams' unique needs and workflows. Automation capabilities for routine tasks can enhance efficiency and consistency.

Leveraging Splunk for Better Observability

An AHEAD partner and leader in the observability space, Splunk helps platform engineering teams identify and mitigate issues faster and more reliably – all while providing better control over data and costs. Splunk Observability works to improve digital resilience by:

- **Unifying Visibility:** Splunk provides IT Operations and Engineering teams with shared data and workflows, enabling better collaboration and visibility between functions.
- **Accelerating Troubleshooting:** Developers can diagnose problems more accurately and efficiently across any environment to rapidly identify solutions.
- **Controlling Data & Costs:** Organizations can achieve better data flexibility by instrumenting everything in your environment while only paying for what you need.

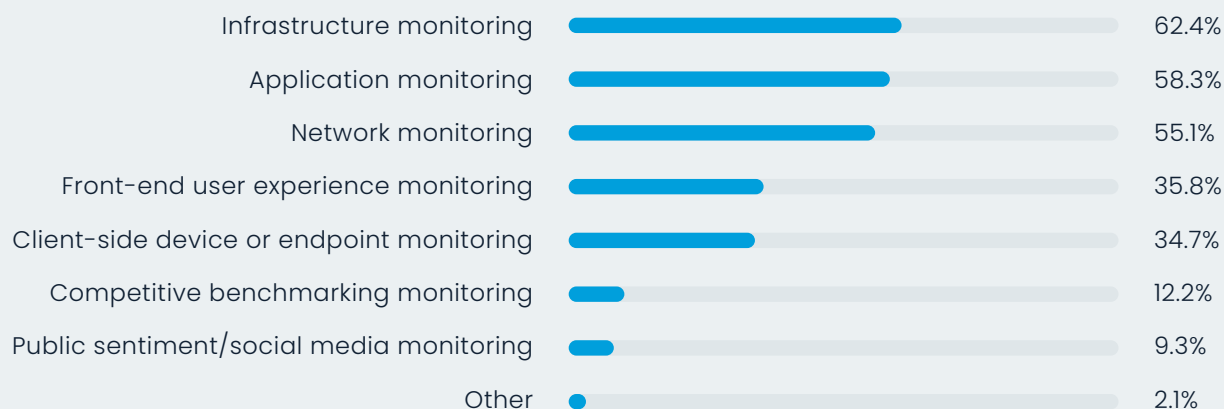
Introduction to Site Reliability Engineering

Site Reliability Engineering (SRE) started at Google in the 2000s and blends software development with operations, focusing on automating tasks and ensuring system reliability. SRE seeks to streamline operations, keeping systems stable and efficient.

The evolution of SRE has seen the adoption of practices like error budgets and Service Level Objectives (SLOs) to balance the need for innovation and stability, influencing how companies manage and operate their IT systems globally.

The [2023 SRE Report](#) by Catchpoint and Blameless highlighted that 54% of organizations use three or more tools for IT operations telemetry, indicating (as shown below) a diverse tool landscape in SRE.

Which telemetry feeds your monitoring or observability frameworks?



[Source](#)

Key Principles of SRE

The key principles of Site Reliability Engineering include:

- **Service Level Objectives (SLOs):** Defines specific, measurable service attributes like availability, latency, or throughput, setting clear standards for the service users can expect.
- **Error Budgets:** Quantifies the acceptable risk or failure rate that a service can handle while still meeting its SLOs. It helps teams balance new feature releases with maintaining service reliability.
- **Release Engineering:** Supports developers and release engineers in reliably deploying new services and features using tools for consistent, repeatable release processes.
- **Monitor Distributed Systems:** Focuses on identifying and resolving system issues by tracking latency, traffic, errors, and saturation metrics to maintain system health and performance.
- **Automation & Tooling:** Automates manual tasks like deployments and fixing failures, making processes more efficient and reliable and freeing up engineers to concentrate on the bigger picture.

The Role of SRE in Platform Engineering

In platform engineering, SRE plays a critical role by:

Ensuring Reliability at Scale

SRE teams work on keeping platforms reliable as they grow. They use scalable tools and methods to spot and fix issues before users are affected, ensuring the platform stays up and running smoothly, even when lots of people are using it.

Collaborating with Development Teams

SRE works closely with development teams to embed reliability into the product development lifecycle. By sharing knowledge and practices, they ensure that reliability considerations are integrated from the outset, leading to more stable and robust systems.

Balancing Innovation with Stability

SRE supports organizations in managing the introduction of new features while keeping systems stable. By using tools like error budgets, SRE offers a structure for taking calculated risks and making sure that new developments don't risk the platform's dependability.

According to [Gartner](#), 75% of enterprises will use SRE practices to optimize product design, cost, and operations.

How Observability & Site Reliability Engineering Work Together

Observability plays an integral role in SRE, serving as the foundation for understanding, maintaining, and enhancing system performance and reliability. Let's take a look at the several key reasons why observability is crucial in SRE:

Problem Detection & Diagnosis

Observability gives SRE teams the tools they need to understand system operations, allowing them to find and fix issues early and preventing major problems, such as outages. Observability leaders are reportedly [2.1 times more likely](#) to detect problems in internally-developed applications within minutes.

Maintaining System Reliability

With thorough observability, SRE teams can deeply understand and monitor their systems for reliable performance. By spotting and fixing issues early, they can boost system stability and reduce unexpected problems.

Rapid Resolution of Issues

Observability allows engineers to quickly find and solve the root causes of issues, lessening their effect on system performance. This fast response is key to keeping systems up and running smoothly and ensuring that users have a good experience.

Preventing Overload & Burnout

Observability gives SRE teams the insight to sort their tasks by urgency, tackling the most critical issues first. This approach is vital for managing the demanding workload of SRE teams and preventing burnout by making workloads more manageable.

Fostering Continuous Improvement

Observability gives detailed insights into system performance, helping SREs spot patterns and areas to improve. This ongoing analysis supports continuous improvement initiatives, leading to more robust and efficient systems.

Enhancing Customer Satisfaction

Quickly addressing and resolving issues not only benefits the internal team, but also enhances the end-user experience. By ensuring applications and services run smoothly, SREs contribute to positive customer experiences, which are critical for competitive advantage and business success.





Key Considerations for Platform Engineering

Having covered the functions and impact of observability and SRE on platform engineering, let's explore a few best practices that every organization should strive for.

Aim for 100% Observability

Setting objectives for complete observability is crucial in platform engineering. Here's a refined approach to embedding observability into every layer of your infrastructure and applications:

- **Data Security & Observability:** Start with a security-first mindset. Before collecting any data, evaluate its sensitivity. Assess whether the monitoring data contains sensitive information that could compromise user privacy or security, and ensure access controls are appropriately stringent.
- **Database Monitoring:** When configuring databases, it's essential to monitor not just the basics, like CPU and memory usage, but also database-specific metrics, such as tablespace health, buffer and cache efficiency, and the dynamics of connections and locks. Tailor your monitoring to the specific needs and peculiarities of the database.
- **Kubernetes Observability:** For Kubernetes environments, automate data collection related to nodes, pods, containers, and logs right from the start. This comprehensive data capture enables a deep understanding of your cluster's operation and performance.
- **Application Performance Monitoring (APM):** Incorporate APM into your application servers to collect detailed performance data. This should include application logs and metrics that reflect the server's operational health.
- **External Service Monitoring:** Don't forget to monitor external dependencies like message queues and web servers. These components are vital to your application's overall performance and reliability.

Automate Everything

To fully embrace automation in platform engineering, integrating observability into your continuous integration and continuous deployment (CI/CD) processes is a critical next step. Here's a streamlined approach to achieving this:

- **Implement Default Alerts Based on Golden Signals:** Start by setting up default alerts focusing on the “golden signals” of monitoring: latency, traffic, errors, and saturation. These signals provide a solid foundation for understanding the health of your system.
- **Create Generic Dashboards with Strategic Tagging:** Develop generic dashboards that utilize tags or naming conventions to automatically aggregate metrics aligned with key performance indicators (KPIs). This method offers an immediate overview of system health and performance, enabling quick responses to emerging issues.
- **Automate Alert & Dashboard Deployment:** Use [REST API](#) or GraphQL for alert and dashboard deployment. Automation in this area ensures that as new services are developed and deployed, they are immediately covered by your observability framework.
- **Maintain Continuous Enhancement of Observability Practices:** Adopt a continuous improvement mindset toward your observability practices. Regularly review and refine your alerts and dashboards to remain relevant and effective.





Final Thoughts

Integrating observability and SRE into platform engineering presents a strategic method to address the complexity and demands of modern software systems. These practices offer a framework for developing, managing, and optimizing scalable, reliable platforms that support continuous innovation and deliver exceptional user experiences.

Observability helps you stay ahead by keeping tabs on system health and how well things are running. On the other side, SRE makes sure your online services work well and remain stable – even as new features are added. Working together, they empower teams to work more efficiently, rapidly deliver new apps to users, and ensure that everything scales up safely and securely. This combination makes sure you're always ready to handle whatever comes next with confidence and reliability.

Contributing Author:

Stephen D. King, Chief Architect, Digital Solutions

AHEAD

Combining cloud-native capabilities in software and data engineering with an unparalleled track record of modernizing infrastructure, we're uniquely positioned to help accelerate the promise of digital transformation.

[Contact Us](#)

National Hubs

CHICAGO

444 W. Lake Street
Suite 3000
Chicago, IL 60606

NEW YORK

500 Fifth Avenue
Suite 1500
New York, NY 10110

ATLANTA

1117 Perimeter Center
Suite W406
Atlanta, GA 30338

SAN FRANCISCO

2000 Crow Canyon Place
Suite 250
San Ramon, CA 94583