Timeline of Host Monitoring Tools

Zeynep Öztürk, Tim Betzer*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: zeynep.oeztuerk@tum.de, betzer@net.in.tum.de

***Abstract*—Network monitoring is a fundamental component of ensuring the reliability, performance, and security of computer networks.**

This paper traces the evolution of host monitoring tools, from early packet-switching networks to modern observability platforms. It highlights the increasing complexity of network environments and the corresponding need for advanced monitoring solutions. The paper explores the integration of monitoring tools in cloud-native infrastructures, with particular attention to Kubernetes orchestration. Prometheus, a tool with a pull-based architecture, and Nagios, a widely-used legacy system, are examined as case studies to illustrate the trade-offs between traditional and modern approaches. The paper concludes by discussing current challenges in network observability, such as automation, real-time alerting, and the role of open-source tools in contemporary IT infrastructures.

***Index Terms*—network monitoring, host monitoring, SNMP, Prometheus, Nagios, Kubernetes, network management, network performance monitoring**

1. Introduction

Since the introduction of the first computer networks, the need for monitoring and managing them has been a critical aspect of ensuring performance and reliability. Network monitoring began in the 1960s and 1970s with simple packet-switching systems and basic command-line utilities. These tools enabled administrators to check device statuses and troubleshoot connectivity issues.

As networks expanded in size and complexity, it became clear that more sophisticated monitoring solutions were needed. Early tools, often developed in-house, were limited in functionality and required substantial manual effort. Despite their limitations, they laid the foundation for the development of more advanced systems.

The 1980s marked the emergence of commercial network monitoring tools, which were generally proprietary, expensive, and used by large organizations. These tools provided a more comprehensive view of network performance, helping administrators address bottlenecks and faults more effectively.

A significant milestone was the introduction of the Simple Network Management Protocol (SNMP) in the late 1980s. SNMP offered a standardized method to monitor and manage devices across different vendors, simplifying the deployment and maintenance of monitoring infrastructures. This standardization supported the rise of both commercial and open-source monitoring tools.

This paper provides a comprehensive overview of the evolution of host monitoring tools, beginning with traditional methods and continuing to modern, cloud-native observability platforms. It examines the fundamental Push vs. Pull data collection models and presents Prometheus and Nagios as case studies to explore the trade-offs between legacy and modern solutions. The discussion concludes with an analysis of current trends, such as real-time alerting, automation, and the growing importance of open-source tools in today's IT environments.

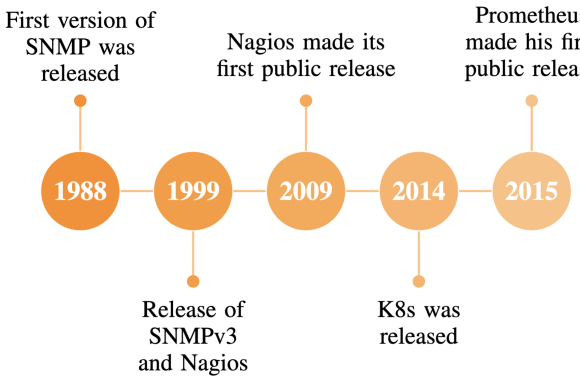

Figure 1: Timeline of host monitoring tools

2. Push vs. Pull

The general principle for collecting data in monitoring systems is divided into two primary models: the **Push Model** and the **Pull Model**.

2.1. Push Model

In the push model, monitored devices send their status information autonomously to the monitoring system. This typically occurs through agents or log forwarding and is well-suited for continuous management.

An example of a tool following this model is Checkmk, which uses agents to transmit data.

2.2. Opportunities and Challenges of the Push Model

One of the key advantages of the push model is the ability to detect problems quickly, as devices transmit alerts as soon as an issue arises. This is particularly beneficial in environments where real-time monitoring

 doi: 10.2313/NET-2025-11-3_09

is essential, such as financial institutions or healthcare systems. Additionally, the push model can help reduce overall network load since data is not polled constantly.

However, the push model presents certain drawbacks. It generally requires additional configuration effort, such as deploying and maintaining agents on the monitored devices, which can be time-consuming and operationally complex. Moreover, since data is transmitted continuously, this approach can lead to increased storage demands. This is especially problematic in environments with limited storage capacity or strict data retention policies.

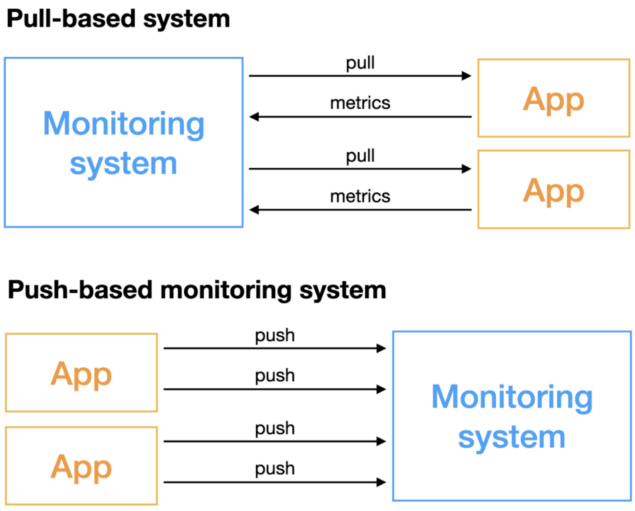

Figure 2: Push vs Pull
Source: [1]

2.3. Pull Model

In contrast, the pull model is based on the principle that the monitoring system initiates data collection by querying the monitored devices at regular intervals. This follows the request–response paradigm and is commonly implemented using protocols such as SNMP, WMI, or API-based interfaces. The pull model is particularly suitable for ad hoc monitoring and manual, on-demand analysis.

A practical example is Nagios, where the monitoring server polls the status of devices—such as servers or network hardware—at fixed intervals, typically every five minutes [2].

2.4. Opportunities and Challenges of the Pull Model

The pull model offers several advantages. It gives the central monitoring system control over the frequency of data collection, which can help balance network load and optimize system performance. Another benefit is the reduced complexity in deployment, as it does not require agents to be installed on the monitored devices.

On the downside, the pull model can introduce latency in problem detection. Since data is only collected at scheduled intervals, issues may go unnoticed until the next polling cycle. This delay can be problematic in environments that require real-time responsiveness, such as financial or healthcare sectors [3].

3. SNMP

The first attempts to create a unified monitoring tool date back to the 1960s and 1970s, when it became evident that the number of hosts in computer networks was growing rapidly [4]. During that time, monitoring systems were mostly proprietary, tailored to specific hardware and vendor environments.

The development of the **Simple Network Management Protocol** (SNMP) in the 1980s by a group of university researchers marked a turning point. It introduced a standardized framework for monitoring and managing devices across heterogeneous networks.

Although SNMP was originally intended as a temporary solution—"to fill the need for a network management tool while a more theoretically sound system was being developed by the IAB" [4]—it ultimately became the de facto standard, and the intended transition never occurred.

The first version of SNMP was released in 1988, with a design philosophy focused on simplicity and ease of implementation. In 1993, **SNMPv2** was introduced, bringing enhancements such as bulk data retrieval and improved error handling. However, it still relied on community strings for authentication, which were considered insecure. In response to these concerns, **SNMPv3** was released in 1999, offering robust security features such as authentication, encryption, and message integrity [4].

Today, SNMPv3 is the standard protocol for monitoring and managing network devices, including switches, routers, firewalls, printers, and servers.

3.1. Evaluation

One of the key factors behind SNMP's widespread adoption is its open nature. It is free software, not controlled by any specific vendor, allowing organizations to implement or build SNMP-based tools without incurring licensing costs or facing vendor lock-in. This has led to a broad ecosystem of SNMP-compatible devices and software solutions.

This openness is especially valuable in network management, where organizations often deploy heterogeneous environments with equipment from various vendors. SNMP provides a unified framework for managing such diverse devices, enabling centralized visibility and control [5].

Although SNMP has not demonstrated significant shortcomings in fault and performance management since its introduction, its continued prevalence can also be attributed to the complexity involved in migrating to alternative protocols or platforms.

Nevertheless, SNMP does have certain limitations. Being a pull-based protocol, it relies on the management system to periodically query devices for information. This approach can cause performance bottlenecks in large-scale environments, where thousands of devices may need to be polled regularly. Moreover, SNMP lacks robust support for complex data modeling and does not efficiently capture relationships between devices. These factors limit its scalability and adaptability in modern, dynamic network infrastructures, where topologies change frequently and systems must respond in real-time [4], [6], [7].

4. Kubernetes

Kubernetes, commonly known as K8s, is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications across clusters of machines.

4.1. History

The initial public commit of Kubernetes was made on GitHub on June 6, 2014. Shortly afterward, in September of the same year, Google officially announced Kubernetes as the open-source successor to its internal container management system, known as Borg.

From the beginning, several major technology companies, including Microsoft, RedHat, IBM, and Docker, joined the Kubernetes community, contributing to its rapid development and adoption. In 2015, with the release of Kubernetes version 1.0, Google donated the project to the newly established **Cloud Native Computing Foundation** (CNCF).

The CNCF, which operates under the umbrella of the nonprofit Linux Foundation, is a vendor-neutral organization that supports the development and adoption of cloud-native technologies. It serves as a collaborative home for open-source projects that are critical to the cloud-native ecosystem.

By 2016, Kubernetes became the first project to be officially hosted by the CNCF. In 2017, it achieved another milestone by becoming the first project accepted into the CNCF's Incubation Program. The following year, in 2018, Kubernetes was the first project to graduate from the CNCF, signaling its maturity and widespread adoption.

Today, Kubernetes is recognized as the second-largest open-source project in the world, surpassed only by the Linux kernel. It has become the default container orchestration solution used by approximately 71 percent of Fortune 100 companies. Kubernetes has played a pivotal role in shaping modern cloud-native architectures, setting a standard for cloud service providers and transforming how applications are developed, deployed, and managed in distributed environments [8], [9].

As of now, Kubernetes continues to be one of the fastest-growing and most actively maintained open-source projects in software history.

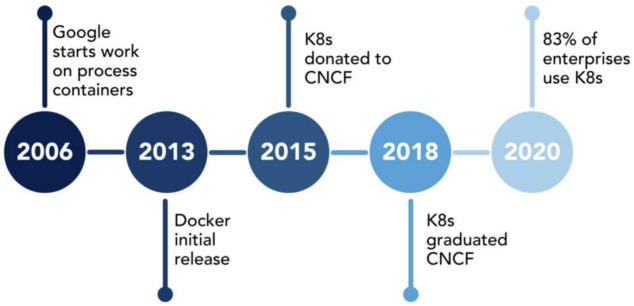

Figure 3: A brief history of Kubernetes
Source: [10]

4.2. Evaluation

Kubernetes offers numerous advantages that have made it the dominant platform in container orchestration. A key benefit is its ability to automatically scale applications based on real-time demand, improving resource efficiency and reducing operational costs. It ensures high availability of services through built-in features such as self-healing, load balancing, and automated failover, thereby enhancing system reliability and minimizing downtime.

Additionally, Kubernetes supports workload portability and seamless integration across hybrid and multi-cloud environments. This allows organizations to deploy and manage applications consistently across different infrastructures, which is particularly beneficial for enterprises with complex IT landscapes. Its robust networking capabilities also simplify service discovery, DNS management, and communication among distributed application components.

Despite its strengths, Kubernetes presents a range of challenges. Its architecture and operational model introduce significant complexity, particularly during initial deployment and cluster setup. The learning curve for administrators and developers new to Kubernetes can be steep, often requiring deep technical understanding and hands-on experience. Running Kubernetes in a production-grade environment demands continuous monitoring, fine-tuning, and specialized skills in areas such as networking, security, and observability.

Moreover, Kubernetes can be resource-intensive, which may not align well with the constraints of small-scale projects or low-powered edge computing environments. Debugging and incident resolution within Kubernetes clusters can also be complex due to the distributed nature of workloads and the limited visibility provided by default tools. In many cases, advanced observability solutions must be integrated to perform effective root cause analysis [8], [9], [11].

5. Prometheus

Started as an internal project at SoundCloud in 2012, **Prometheus** is now a widely used open-source monitoring and alerting toolkit. It focuses specifically on cloud-native environments, where it excels in collecting, storing, and analyzing metrics in real time.

After gaining increasing adoption within SoundCloud, and later from Docker and Boxever in 2014, Prometheus had its first public launch on January 26, 2015. Within one year of open-source development, Prometheus had grown into a thriving community of users and contributors. The project saw significant activity, with over 200 contributors, more than 2,300 pull requests, and over 4,800 stars on GitHub. Even large companies such as Google and CoreOS began using Prometheus in their infrastructure. "Google is now instrumenting its open-source container management system Kubernetes natively with Prometheus metrics," stated Volz [12].

On May 9, 2016, Prometheus joined the Cloud Native Computing Foundation (CNCF) as its second open-source project, following Kubernetes [5].

 doi: 10.2313/NET-2025-11-3_09

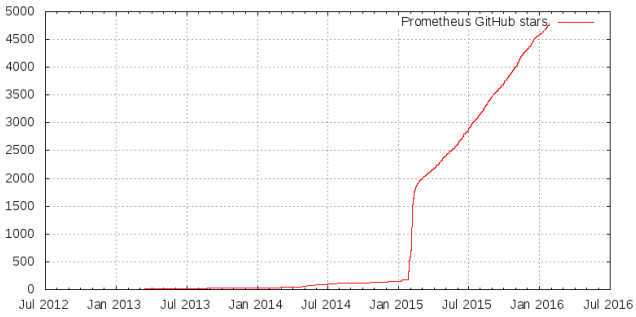

Figure 4: Peak after getting submitted to Hacker News
Source: [13]

5.1. Evaluation

Prometheus is a powerful monitoring and alerting toolkit, well-suited for cloud-native environments. Due to its pull-based architecture, it easily monitors dynamic systems, making it a popular choice for organizations using microservices and container orchestration platforms like Kubernetes. It is designed to be fast and efficient, with a time-series database that enables high-performance data storage and retrieval. It also supports flexible monitoring capabilities, allowing users to define custom metrics and alerts based on their needs. Additionally, it is highly scalable and can manage large volumes of data, making it suitable for organizations with complex infrastructures.

However, it does have some limitations. Prometheus does not provide native long-term storage for metrics, which can be a drawback for organizations that need to retain historical data for compliance or analytical purposes. It also focuses primarily on metrics and lacks built-in support for logs or distributed clustering. This can make it less suitable for organizations that require a more comprehensive monitoring solution that includes logs and distributed tracing [5], [11], [14], [15].

6. Nagios

Nagios is one of the oldest and most widely known monitoring tools for IT infrastructures. It enables the monitoring of servers, networks, applications, and services to detect and respond to issues proactively.

Nagios was developed by Ethan Galstad in 1999 as a side project called NetSaint. It was initially developed as an open-source solution for network monitoring, with a primary focus on hosts and services. Due to a trademark issue with the term “Saint” in 2001, the project’s name was changed to Nagios, an acronym for “Nagios Ain’t Gonna Insist On Sainthood“ [2]. In 2007, Nagios Enterprises was founded, and after two years of development, the first commercial edition of Nagios was released on December 31, 2009.

6.1. Evaluation

Thanks to its plugin-based architecture, Nagios can be extended to monitor a wide range of services and applications, such as the availability and performance of network devices, servers, and applications. It also supports monitoring system metrics including CPU usage, memory consumption, disk space, and network traffic. Widely adopted and well established, “Nagios was awarded ‘Best Monitoring Application’ in the 2013 Linux Journal Readers’ Choice Awards“ [16]—its third time receiving this title.

Despite its capabilities, Nagios is not ideal for large, dynamic environments where services and hosts change frequently. In such cases, the manual configuration requirements of Nagios can become cumbersome and time-consuming [2], [16].

7. Conclusion and future work

Since the beginning of the computer networks and the rising of the complexity of the network systems, the need for monitoring tools has increased. Since then SNMP has been the most widely used protocol for monitoring network devices and Kubernetes has become one of the most popular container orchestration platform. With the Prometheus monitoring system, it is possible to monitor the Kubernetes cluster and the applications running on it. Nagios is a widely used monitoring tool, which is used to monitor the host system.

Although commonly used host monitoring tools perform well for most modern systems, the continuous advancement of technology necessitates the development of newer and more sophisticated monitoring solutions. One emerging area of focus is the monitoring of **quantum computers**, which are gaining popularity and present unique challenges. Unlike classical systems that operate with bits representing either 0 or 1, quantum computers use qubits, which can exist in a superposition of states. This fundamental difference makes monitoring quantum systems significantly more complex than traditional computing environments.

Nevertheless, major companies such as IBM and Amazon are actively working on solutions in this domain. For instance, Qiskit, an open-source quantum computing framework developed by IBM, includes components that support basic monitoring of quantum systems.

As quantum computing technology continues to evolve, the demand for advanced monitoring tools will grow in order to ensure the performance, stability, and reliability of future quantum-based infrastructures [17].

References

[1] “Push vs pull architecture in cloud computing,” https://www.alibabacloud.com/blog/push-vs-pull-architecture-in-cloud-computing_595202, [Online; accessed 6-April-2025].

[2] Nagios, “The nagios story,” https://www.nagios.org/story/, [Online; accessed 30-March-2025].

[3] J.-P. Martin-Flatin, “Push vs. pull in web-based network management,” 1998.

[4] “Snmp history,” https://docs.lextudio.com/snmp/snmp-history, [Online; accessed 1-April-2025].

[5] “The prometheus monitoring system and time series database.” https://www.cncf.io/projects/prometheus/, [Online; accessed 31-March-2025].

[6] M. Fedor, M. Schoffstall, J. R. Davin, and D. J. D. Case, “A simple network management protocol (snmp),” May 1990.

[7] K. Brister, “High tech goes low key,” https://snmp.com/company/history.shtml, 1998, [Online; accessed 1-April-2025].

[8] “Kubernetes project journey report,” 2023, [Online; accessed 28-March-2025].

[9] S. Susnjara, “The history of kubernetes,” https://www.ibm.com/think/topics/kubernetes-history, 2023, [Online; accessed 28-March-2025].

[10] “Who made kubernetes?” https://www.opsramp.com/guides/why-kubernetes/who-made-kubernetes/, [Online; accessed 6-April-2025].

[11] N. S. an Elizabeth Bautista, “Towards a framework for monitoring and analyzing high performance computing environments using kubernetes and prometheus,” IEEE, Ed., 2019.

[12] D. Harrington, B. Wijnen, and R. Presuhn, “An architecture for describing simple network management protocol (snmp) management frameworks,” *RFC 3411*, 2002.

[13] “One year of open prometheus development,” https://prometheus.io/blog/2016/01/26/one-year-of-open-prometheus-development/, [Online; accessed 6-April-2025].

[14] G. Darwesh, “Kubernetes monitoring with prometheus for security purposes,” in *Sammelband wissenschaftlicher Forschungsarbeiten auf Grundlage des internationalen Wettbewerbs, Technologische Innovationen und wissenschaftliche Entdeckungen*, A. A. Vorobeva, Ed.

[15] J. Turnbull, *Monitoring with Prometheus*, September 10, 2018.

[16] J. Gray, “Reader’s choice awards 2009,” *Linux Journal*, June 1, 2009.

[17] “Introduction to qiskit,” https://docs.quantum.ibm.com/guides, [Online; accessed 5-April-2025].