International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 7.101

A Comparative Analysis of System Monitoring Architecture: Evaluating Prometheus, Elk Stack, and Custom dashboards for performance and Scalability

Rahul Banerjee

Abstract: ***The proliferation of microservices and cloud-native architectures has rendered traditional system monitoring approaches insufficient, necessitating a paradigm shift towards comprehensive observability. This paper presents a comparative analysis of three dominant system monitoring architectures: the metrics-centric Prometheus ecosystem, the log-centric ELK (Elastic) Stack, and the flexible paradigm of Custom Dashboards, exemplified by Grafana. We conduct an architectural deep dive into each solution, evaluating their core components, data models, and ingestion mechanisms. The analysis focuses on key performance indicators, including ingestion throughput, query latency, and resource utilization, alongside a rigorous assessment of their scalability models for handling high data volumes, managing metric cardinality, and ensuring long-term data retention. Through an examination of real-world case studies and quantitative benchmarks, this paper establishes that the optimal choice of monitoring architecture is not a matter of universal superiority but a strategic decision contingent on an organization's primary telemetry signals, operational maturity, and specific scalability challenges. Prometheus excels in real-time, high-frequency metric analysis and alerting within dynamic environments, whereas the ELK Stack provides unparalleled capabilities for deep, full-text log analysis and security forensics. Custom Dashboards offer maximum flexibility by decoupling the visualization layer from the data backend, enabling hybrid solutions but demanding greater engineering investment. This paper concludes with a decision-making framework to guide architects and engineers in selecting the most suitable monitoring strategy for their distributed systems and a forward-looking perspective on emerging technologies like OpenTelemetry and eBPF that are reshaping the observability landscape.***

Keywords: System monitoring, observability, Prometheus, ELK Stack, Grafana, custom dashboards, performance metrics, scalability

1. Introduction

The landscape of software architecture has undergone a transformative evolution over the past decade, characterized by a strategic migration from monolithic designs to distributed microservices architectures.[1] In a monolithic system, all application components—user interface, business logic, and data access layers—are tightly coupled and deployed as a single, indivisible unit.[3] This approach, while simple to develop and deploy initially, presents significant challenges as systems grow in complexity. Scaling a monolith is an "all or nothing" proposition, where the entire application must be scaled to meet increased demand on a single component, leading to inefficient resource utilization.[2] Furthermore, the tightly coupled nature of the codebase makes implementing changes cumbersome and risky, dramatically slowing down deployment cycles and stifling innovation.[5]

In response to these limitations, the microservices paradigm emerged, advocating for the decomposition of large applications into a collection of small, autonomous services, each responsible for a specific business function.[1] These services communicate over well-defined APIs and can be developed, deployed, and scaled independently.[3] This architectural style offers profound benefits, including enhanced scalability, as individual services can be scaled based on demand; faster deployment cycles, as smaller, independent updates reduce risk; and improved fault isolation, where the failure of one service does not cascade to the entire system.[1] However, this distribution of components introduces a new set of complex challenges. What was once a simple in-process function call in a monolith becomes a network request in a microservices architecture, subject to latency, unreliability, and new failure modes.[3] Managing data consistency across distributed databases, securing inter-service communication, and debugging requests that traverse multiple services add significant operational overhead.[2] This fundamental shift has rendered traditional monitoring techniques, designed for the simplicity of monolithic systems, inadequate for the distributed and dynamic nature of modern applications.[7]

The inherent complexity of microservices demanded a more profound approach to system insight than what traditional monitoring could provide. Monitoring, in its classic sense, focuses on tracking a predefined set of metrics and logs to identify "known unknowns"—predictable failure modes such as high CPU usage or low disk space.[4] This approach works well for stable, monolithic systems where the potential failure states are well understood. However, in a distributed system, the number of interconnected parts and potential failure modes grows exponentially, leading to a prevalence of "unknown unknowns"—emergent, unpredictable problems that cannot be anticipated and pre-monitored.[8]

This challenge gave rise to the concept of observability. Originating from control theory, observability is defined as the ability to infer the internal states of a system by examining its external outputs.[8] In the context of IT systems, this means instrumenting applications to produce rich, high-fidelity telemetry data that allows engineers to ask arbitrary questions about system behaviour and diagnose unforeseen issues without needing to ship new code.[4] This practice is built upon three primary data types, often referred to as the "three pillars of observability" [11]:

1) **Metrics:** Numerical, time-stamped measurements of system health and performance, such as request counts, latency, and resource utilization. Metrics are optimized for storage and mathematical aggregation, making them ideal for dashboards, alerting, and trend analysis.[11]
2) **Logs:** Granular, time-stamped, and immutable records of discrete events. Logs provide detailed, contextual information about what an application was doing at a specific point in time, making them invaluable for debugging and root cause analysis.[10]
3) **Traces:** A representation of the end-to-end journey of a single request as it propagates through multiple services in a distributed system. Traces are essential for understanding inter-service dependencies, identifying performance bottlenecks, and visualizing the flow of operations.[10]

Together, these three pillars provide a holistic view of a system's health, enabling teams to move from a reactive posture of fixing known problems to a proactive one of exploring and understanding complex system dynamics.[8]

2. Architectural Deep Dive: The Elk Stack

The ELK Stack, now officially known as the Elastic Stack, is a powerful suite of open-source tools designed for centralized log management, search, and real-time data analysis.[50] While Prometheus is fundamentally a metrics-first system, the ELK Stack is a log-first architecture, built around the formidable search capabilities of its core component, Elasticsearch.

1) Core Components and Data Flow Pipeline

The Elastic Stack is composed of four primary components that form a comprehensive data pipeline [50]:

a) **Elasticsearch:** The heart of the stack, Elasticsearch is a distributed, RESTful search and analytics engine built on Apache Lucene. It is responsible for storing, indexing, and making vast quantities of data searchable in near real-time.[50]
b) **Logstash:** A server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it to a "stash" like Elasticsearch.[53]
c) **Kibana:** The visualization layer of the stack. Kibana provides a web-based user interface for exploring, visualizing, and discovering insights from the data stored in Elasticsearch. It allows users to create dashboards with charts, graphs, maps, and tables.[50]
d) **Beats:** A family of lightweight, single-purpose data shippers. Beats are installed on servers to collect and forward specific types of data to either Logstash or directly to Elasticsearch.[53]

The canonical data flow pipeline begins with Beats collecting data at the source. This data is then sent to Logstash for parsing, enrichment, and transformation. Finally, the processed data is indexed in Elasticsearch, where it becomes available for search and visualization in Kibana.[51] This pipeline is highly flexible; for simpler use cases, Beats can ship data directly to Elasticsearch, bypassing Logstash entirely.[56]

2) The Push-Based Ingestion Model with Beats and Logstash

Unlike Prometheus's pull model, the ELK Stack operates on a **push-based model**, where agents on the monitored systems actively send data to the central processing and storage components.

Beats are the primary agents for this task. They are designed to be lightweight with a minimal resource footprint, making them safe to deploy on production servers.[58] The Beats family includes several specialized shippers [56]:

a) **Filebeat:** Tails log files and forwards log events.
b) **Metricbeat:** Collects system and service metrics (e.g., from Docker, Nginx).
c) **Packetbeat:** Monitors network traffic and decodes application-layer protocols.
d) **Auditbeat:** Collects audit data about system and file integrity changes.

These agents handle the initial collection and forwarding of data. For more complex processing, the data is sent to **Logstash**. Logstash is a powerful and flexible ETL (Extract, Transform, Load) tool. Its pipeline architecture consists of three stages: inputs, filters, and outputs.[60]

- **Inputs:** Logstash supports a wide array of input plugins to ingest data from sources like files, Beats, message queues (e.g., Kafka), and databases.[62]
- **Filters:** This is where the core data processing happens. Logstash provides a rich set of filter plugins to parse, structure, and enrich the data. The grok filter is commonly used to extract structured fields from unstructured log text using regular expressions. The mutate filter can add, remove, or modify fields, and the geoip filter can enrich log data with geographical information based on an IP address.[60]
- **Outputs:** After processing, Logstash sends the data to one or more destinations using output plugins, with Elasticsearch being the most common target.[61]

In high-volume environments, a direct connection from data shippers to Logstash or Elasticsearch can be risky. A sudden spike in log volume can overwhelm the processing pipeline, leading to data loss.[64] To mitigate this, a best practice is to introduce a durable buffer, such as Apache Kafka or Redis, between the shippers and Logstash. This buffer absorbs traffic spikes and ensures data persistence even if downstream components are temporarily unavailable.[54]

3) Performance Characteristics: Elasticsearch and Full-Text Indexing

The performance and core functionality of the ELK Stack are defined by **Elasticsearch**. It is a document-oriented NoSQL database where data is stored as JSON documents.[53] Unlike a traditional database, Elasticsearch's primary strength lies in its search capabilities, which are powered by an underlying data structure called an **inverted index**.[67]

An inverted index is a mapping from content, such as words or numbers, to their locations in a set of documents. Instead of searching documents sequentially, Elasticsearch looks up the search term in the inverted index to get a list of documents that contain it, enabling extremely fast full-text searches.[69] During the indexing process, Elasticsearch analyzes the text fields of a document, breaking the text into individual terms

(tokenization) and normalizing them (e.g., converting to lowercase) before adding them to the inverted index.[69]

This "index everything" approach is what gives the ELK Stack its power for log analysis. It allows users to perform complex, ad-hoc queries on unstructured log data with very low latency. However, this power comes at a significant cost. Creating and maintaining inverted indexes for every field is resource-intensive, leading to a much larger storage footprint and higher CPU and memory consumption compared to a time-series database like Prometheus, which only indexes labels.[71]

Data exploration is primarily done through **Kibana**, which provides a user-friendly interface for building queries, creating visualizations, and assembling dashboards.[74] Users can interact with their data using the Kibana Query Language (KQL), a simple text-based query language, or by directly writing more complex queries using the Elasticsearch Query DSL or Lucene syntax.[75]

4) Scalability Architectures for High-Volume Log Management

Elasticsearch is inherently designed as a distributed system, built for horizontal scalability from the ground up.[53] Its scalability model revolves around a few key concepts [66]:

- **Cluster:** A collection of one or more nodes that work together.
- **Node:** A single server that is part of a cluster. Nodes can have different roles, such as **master nodes** (responsible for managing the cluster state) and **data nodes** (responsible for storing data and executing queries).[53] For stability in production environments, it is a best practice to have dedicated master nodes.[64]
- **Index:** A collection of documents with similar characteristics, analogous to a database in a relational system.
- **Shard:** Because an index can grow to hold a massive amount of data that exceeds the hardware limits of a single node, Elasticsearch allows an index to be subdivided into multiple pieces called shards. Each shard is a fully functional and independent index that can be hosted on any data node in the cluster.[66] Sharding allows for horizontal scaling of both storage capacity and query throughput, as operations can be parallelized across shards.
- **Replica:** A copy of a shard. Replicas provide redundancy and high availability; if a node containing a primary shard fails, a replica shard on another node can be promoted to become the new primary.[53] Replicas also improve read performance, as search requests can be handled by either the primary or replica shards.

To scale an Elasticsearch cluster, administrators can simply add more data nodes. Elasticsearch will automatically rebalance the shards and replicas across the available nodes to distribute the load.[77] However, managing a large-scale cluster requires careful planning. The number of primary shards for an index is fixed at creation time and cannot be changed, so it must be chosen carefully based on anticipated data volume. A common scaling bottleneck is not the data itself, but the **cluster state**, which is the metadata about all nodes, indices, shards, and mappings. An excessively large cluster state, often caused by having too many shards or too many unique fields in mappings, can overwhelm the master node and lead to cluster instability.[71]

For managing long-term data retention and storage costs, Elasticsearch offers **index lifecycle management (ILM)** and **data tiers**. ILM policies can be defined to automatically manage indices as they age. For example, an index can be moved from a high-performance "hot" tier (using fast SSDs) to a less expensive "warm" tier, and eventually to a low-cost "cold" or "frozen" tier (using slower spinning disks or object storage) before being deleted.[53] This tiered approach allows organizations to retain massive volumes of log data cost-effectively.

3. Comparative Analysis: Performance and Scalability

The selection of a monitoring architecture is a critical engineering decision with long-term implications for system reliability, operational efficiency, and cost. This section provides a direct comparison of the Prometheus ecosystem and the ELK Stack, focusing on their fundamental performance characteristics and scalability models. The fundamental trade-off between these systems stems from their core design philosophies. The ELK Stack's "index everything" approach provides incredible query flexibility for unstructured text by pre-computing the locations of all terms. This comes at the cost of massive storage and compute overhead, as every log line is fully parsed and indexed.[71] Prometheus, in contrast, is optimized for numerical aggregation. Its indexing is focused on rapidly finding and grouping time series by labels, not on searching the content of the values themselves. This makes it highly efficient for its intended purpose but unsuitable for the deep log analysis that is ELK's strength.

Table I: Core Architectural Characteristics Comparison

Feature	Prometheus Ecosystem	ELK (Elastic) Stack
Primary Use Case	Real-time metrics monitoring and alerting [102]	Centralized log management and full-text search[76]
Data Model	Dimensional Time Series (metric name + labels) [18]	Document-oriented (JSON documents) [53]
Core Telemetry Signal	Metrics (numerical time series) [103]	Logs (unstructured/structured text) [103]
Collection Model	Pull-based (server scrapes targets) [22]	Push-based (agents push data) [103]
Query Language	PromQL (functional, for time-series) [33]	KQL / Lucene / Query DSL (for full-text search)[68]
Default Visualization	Basic Expression Browser (Grafana is standard) [73]	Kibana (integrated) [50]
Scalability Model	Single-node with ecosystem extensions (Thanos/Cortex)[37]	Natively distributed (cluster of nodes) [73]

a) Performance Evaluation: Ingestion Throughput, Query Latency, and Resource Footprint

Prometheus is engineered for high-frequency, high-throughput ingestion of numerical data. Its TSDB is optimized for fast writes and appends, allowing a single, well-provisioned instance to ingest hundreds of thousands of

samples per second.[37] Query latency for typical time-series aggregations (e.g., calculating rates, averages, or percentiles over recent time windows) is very low, often in the sub-second range, which is critical for real-time alerting and dashboarding.[104] The resource footprint (CPU, memory, disk) of Prometheus is relatively low for its ingestion rate, but it is extremely sensitive to metric cardinality. As the number of unique time series grows, memory usage for the index can increase dramatically, becoming the primary performance bottleneck.[30] Real-world deployments, such as Uber's M3 platform (a Prometheus-compatible backend), demonstrate the potential for this architecture to scale to massive ingestion rates, handling 500 million metrics per second and storing 6.6 billion time series.[106]

The **ELK Stack**, by contrast, is optimized for ingesting and indexing large volumes of unstructured or semi-structured text data. Elasticsearch can handle very high ingestion throughput, but the process is more resource-intensive than in Prometheus due to the overhead of full-text indexing.[73] For queries involving full-text search, Elasticsearch is exceptionally fast, leveraging its inverted index to return results from petabytes of data in seconds.[68] However, for the kind of complex numerical aggregations that are trivial in PromQL, Elasticsearch can be slower and more computationally expensive. The resource footprint of an ELK cluster is significantly higher than a Prometheus instance handling a comparable number of data points. The JVM heap, disk I/O for indexing, and storage space required for the inverted indexes all contribute to substantial hardware requirements.[64] The scale of Netflix's deployment, with 700-800 production nodes across 100 clusters, exemplifies the infrastructure investment required to support log analytics for a global, high-traffic service.[108]

b) Scalability Evaluation: Data Volume, Cardinality, and Long-Term Retention

A standalone **Prometheus** server is not designed for long-term, scalable storage. Its local TSDB is typically configured with a short retention period (e.g., 15 days) to manage disk space.[27] True scalability for data volume and long-term retention is achieved through the ecosystem of remote storage solutions like Thanos and Cortex. These solutions offload historical data to cost-effective object storage, providing virtually unlimited retention while introducing additional architectural and operational complexity.[43] The primary scalability constraint for Prometheus remains cardinality. Uncontrolled growth in the number of time series can overwhelm even a scaled deployment, necessitating careful metric and label design.[30]

The **ELK Stack** is natively designed for horizontal scalability to handle massive data volumes. By adding more data nodes to an Elasticsearch cluster, both storage capacity and processing power can be scaled out linearly.[53] With features like ILM and data tiers, ELK can manage petabyte-scale deployments and retain data for years in a cost-effective manner.[73] While not susceptible to cardinality in the same way as Prometheus, Elasticsearch has its own scaling challenge: cluster state management. An excessive number of indices, shards, or unique field mappings can bloat the cluster state, which must be managed by the master node and propagated to all other nodes. A large cluster state can slow down cluster operations and become a bottleneck to further scaling.[71]

Custom Dashboards powered by Grafana have no inherent scalability model; their scalability is entirely a function of the chosen data backend. Grafana itself is stateless and can be easily scaled horizontally behind a load balancer. However, it cannot solve the underlying scalability challenges of its data sources. A custom solution using Grafana forces an organization to consciously choose its scalability trade-offs by selecting the appropriate backend for each data type.

Table II: Scalability Solutions and Trade-offs

Feature	Prometheus (Federation)	Prometheus (Thanos)	Prometheus (Cortex)	ELK Stack (Clustering & Data Tiers)
Global Query View	Limited (Aggregated Data) [40]	Yes (Full Fidelity) [44]	Yes (Full Fidelity) [48]	Yes (Native) [53]
Long-Term Storage	No [39]	Yes (Object Storage) [43]	Yes (Object Storage) [47]	Yes (Data Tiers) [71]
High Availability	No (Single Point of Failure)[42]	Yes (Deduplication) [43]	Yes (Replication) [48]	Yes (Replicas) [79]
Multi-tenancy	No	Limited	Yes (Native) [48]	Limited (Spaces/RBAC) [68]
Operational Complexity	Low	Medium	High	High

c) Operational Complexity and Ecosystem Maturity

For initial setup and basic use cases, **Prometheus** is often considered simpler and more lightweight. Its single-binary design and straightforward configuration make it easy to get started.[76] The ecosystem is mature and vast, particularly within the cloud-native community, with hundreds of official and community-contributed exporters available for nearly every common piece of software and hardware.[25] However, as scaling requirements grow, the operational complexity increases significantly. Deploying and managing a distributed system like Thanos or Cortex on top of Prometheus requires deep expertise in distributed systems.[48] The case study of DigitalOcean's migration from Graphite and OpenTSDB to Prometheus highlights this duality. They were drawn to Prometheus for its superior query language and the empowerment it gave developers to create their own metrics and alerts, but they also found it necessary to build their own tooling to manage Prometheus at scale and eventually developed a separate solution for long-term storage.[110]

The **ELK Stack** has a higher initial setup complexity due to its multiple components (Beats, Logstash, Elasticsearch, Kibana) that must be configured to work together.[102] Managing a large Elasticsearch cluster is a non-trivial task, requiring expertise in JVM tuning, shard allocation, and capacity planning to maintain performance and stability.[71] However, because it is an integrated stack from a single vendor (Elastic), the components are designed to work together, and the ecosystem is mature with a strong focus on enterprise-grade features, including security, machine

learning, and extensive support for various log-based use cases like Security Information and Event Management (SIEM).[68]

4. Conclusion

This analysis has demonstrated that Prometheus, the ELK Stack, and Custom Dashboards represent distinct architectural philosophies tailored to different facets of system observability. There is no single "best" solution; rather, each architecture presents a unique set of trade-offs in performance, scalability, and operational complexity. **Prometheus** and its ecosystem are purpose-built for the world of metrics. Its pull-based model, dimensional data structure, and powerful PromQL make it exceptionally effective for real-time monitoring, high-frequency alerting, and performance analysis of dynamic, cloud-native systems. While a single Prometheus instance is operationally simple and highly reliable, its scalability for long-term storage and global query capabilities depends on a complex ecosystem of tools like Thanos or Cortex. Its primary performance constraint is high cardinality, which requires disciplined metric design. The **ELK Stack** is the definitive solution for log-centric observability. Its push-based pipeline and the powerful inverted index of Elasticsearch provide unparalleled capabilities for full-text search, unstructured data analysis, and deep-dive troubleshooting. It is natively designed for horizontal scalability and can manage petabyte-scale data volumes for long-term retention. However, this power comes at the cost of significant resource consumption and higher operational complexity, particularly in managing large Elasticsearch clusters. The long-term success of architectures like Prometheus and the ELK Stack will depend on their ability to integrate seamlessly with these emerging standards and to provide powerful, intelligent analysis layers. The future of system monitoring will likely be characterized not by a single monolithic tool, but by interoperable, hybrid platforms that combine the strengths of different specialized backends under a unified, AI-enhanced analytical framework.

References

[1] The Evolution from Monolithic to Microservices Architecture | Eagle Eye, accessed September 13, 2025, https://eagleeye.com/blog/the-evolution-from-monolithic-to-microservices-architecture
[2] Evolution of Software Architecture: From Monoliths to Microservices and Beyond - DZone, accessed September 13, 2025, https://dzone.com/articles/evolution-of-software-architecture-from-monoliths
[3] Monolith vs. Microservices: A Journey Through Architecture, History, and Evolution, accessed September 13, 2025, https://corner.buka.sh/monolith-vs-microservices-a-journey-through-architecture-history-and-evolution/
[4] Monitoring to Observability: Evolution from Monoliths to Cloud-Native Microservices | by ProtonsAI : Tools for Engineering Excellence | Medium, accessed September 13, 2025, https://blog.protons.ai/monitoring-to-observability-evolution-from-monoliths-to-cloud-native-microservices-a5e26db8d54f
[5] Monolith to Microservices Migration: Guide for Modernizing Enterprise Applications - Neontri, accessed September 13, 2025, https://neontri.com/blog/monolith-microservices-migration/
[6] Monolith Reversion: from Microservices back to Monolith - Ptidej Team Blog, accessed September 13, 2025, https://blog.ptidej.net/monolith-reversion-from-microservices-back-to-a-monolith/
[7] Microservices Monitoring: Challenges, Metrics & Tips for Success - Lumigo, accessed September 13, 2025, https://lumigo.io/microservices-monitoring/
[8] What is Observability? An Introduction - Splunk, accessed September 13, 2025, https://www.splunk.com/en_us/blog/learn/observability.html
[9] What is observability? Not just logs, metrics, and traces - Dynatrace, accessed September 13, 2025, https://www.dynatrace.com/news/blog/what-is-observability-2/
[10] What Is Observability? | IBM, accessed September 13, 2025, https://www.ibm.com/think/topics/observability
[11] Observability in Distributed Systems: Logs, Metrics, and Traces | by Sruthi Sree Kumar | Big Data Processing | Medium, accessed September 13, 2025, https://medium.com/big-data-processing/observability-in-distributed-systems-logs-metrics-and-traces-ee260c60d697
[12] The Three Pillars of Observability: Logs, Metrics, and Traces - CrowdStrike, accessed September 13, 2025, https://www.crowdstrike.com/en-us/cybersecurity-101/observability/three-pillars-of-observability/
[13] Three Pillars of Observability: Logs vs. Metrics vs. Traces | Edge Delta, accessed September 13, 2025, https://edgedelta.com/company/blog/three-pillars-of-observability
[14] Three Pillars of Observability: Logs, Metrics and Traces - IBM, accessed September 13, 2025, https://www.ibm.com/think/insights/observability-pillars
[15] Observability in Distributed Systems - GeeksforGeeks, accessed September 13, 2025, https://www.geeksforgeeks.org/system-design/observability-in-distributed-systems/
[16] The 3 pillars of observability: Unified logs, metrics, and traces | Elastic Blog, accessed September 13, 2025, https://www.elastic.co/blog/3-pillars-of-observability
[17] What is Prometheus Monitoring? A Beginner's Guide | Better Stack Community, accessed September 13, 2025, https://betterstack.com/community/guides/monitoring/prometheus/
[18] Overview - Prometheus, accessed September 13, 2025, https://prometheus.io/docs/introduction/overview/
[19] Prometheus Architecture: A Comprehensive Deep Dive - dev ops, accessed September 13, 2025, https://govi.hashnode.dev/prometheus-architecture-a-comprehensive-deep-dive
[20] What is Prometheus? | New Relic, accessed September 13, 2025, https://newrelic.com/blog/best-

practices/what-is-prometheus
[21] Data model - Prometheus, accessed September 13, 2025, https://prometheus.io/docs/concepts/data_model/
[22] Is Prometheus Monitoring Push or Pull? - SigNoz, accessed September 13, 2025, https://signoz.io/guides/is-prometheus-monitoring-push-or-pull/
[23] Why is Prometheus using a pull model?Blog - O11y, accessed September 13, 2025, https://o11y.eu/blog/prometheus-pull-model/
[24] When to use the Pushgateway - Prometheus, accessed September 13, 2025, https://prometheus.io/docs/practices/pushing/
[25] Prometheus - Monitoring system & time series database, accessed September 13, 2025, https://prometheus.io/
[26] Pull or Push: How to Select Monitoring Systems? - Alibaba Cloud Community, accessed September 13, 2025, https://www.alibabacloud.com/blog/pull-or-push-how-to-select-monitoring-systems_599007
[27] How to Configure and Optimize Prometheus Data Retention - Last9, accessed September 13, 2025, https://last9.io/blog/prometheus-data-retention/
[28] Understanding Time Series Database (TSDB) in Prometheus - Saurabh Adhau's Blog, accessed September 13, 2025, https://devopsvoyager.hashnode.dev/understanding-time-series-database-tsdb-in-prometheus
[29] Storage - Prometheus, accessed September 13, 2025, https://prometheus.io/docs/prometheus/latest/storage/
[30] Optimizing Prometheus Storage: Handling High-Cardinality Metrics at Scale - Medium, accessed September 13, 2025, https://medium.com/@platform.engineers/optimizing-prometheus-storage-handling-high-cardinality-metrics-at-scale-31140c92a7e4
[31] Optimizing Prometheus Logging: Best Practices and Strategies - Graph AI, accessed September 13, 2025, https://www.graphapp.ai/blog/optimizing-prometheus-logging-best-practices-and-strategies
[32] Designing a Metrics and Monitoring System: Prometheus at Scale - DEV Community, accessed September 13, 2025, https://dev.to/sgchris/designing-a-metrics-and-monitoring-system-prometheus-at-scale-1mjo
[33] PromQL tutorial for beginners and humans | by Aliaksandr Valialkin - Medium, accessed September 13, 2025, https://valyala.medium.com/promql-tutorial-for-beginners-9ab455142085
[34] The Beginner's Handbook to PromQL | Better Stack Community, accessed September 13, 2025, https://betterstack.com/community/guides/monitoring/promql/
[35] Query examples | Prometheus, accessed September 13, 2025, https://prometheus.io/docs/prometheus/latest/querying/examples/
[36] An Intro to PromQL: Basic Concepts & Examples - Logz.io, accessed September 13, 2025, https://logz.io/blog/promql-examples-introduction/
[37] Scaling Prometheus: Handling Large-Scale Deployments | by Platform Engineers - Medium, accessed September 13, 2025, https://medium.com/@platform.engineers/scaling-prometheus-handling-large-scale-deployments-ec130e0b7ba8
[38] M3db cluster as a Prometheus long term storage | by Sayf Eddine HAMMEMI, accessed September 13, 2025, https://sayfeddinehammemi.medium.com/m3db-cluster-as-a-prometheus-long-term-storage-dfbbb1f6aeb8
[39] Federation - Prometheus, accessed September 13, 2025, https://prometheus.io/docs/prometheus/latest/federation/
[40] Prometheus Federation Scaling Prometheus Guide - Last9, accessed September 13, 2025, https://last9.io/blog/prometheus-federation-guide/
[41] Implementing Hierarchical Federation With Prometheus - Pluralsight, accessed September 13, 2025, https://www.pluralsight.com/labs/aws/implementing-hierarchical-federation-with-prometheus
[42] Observability | Best Practices for Centralized Data Management of Multiple Prometheus Instances - Alibaba Cloud Community, accessed September 13, 2025, https://www.alibabacloud.com/blog/observability-%7C-best-practices-for-centralized-data-management-of-multiple-prometheus-instances_601178
[43] Scaling Prometheus Using Thanos - OpsRamp, accessed September 13, 2025, https://www.opsramp.com/guides/prometheus-monitoring/prometheus-thanos/
[44] Scaling Prometheus with Thanos: A Guide to Long-Term, Scalable Monitoring - Medium, accessed September 13, 2025, https://medium.com/@Nitish_Mane/scaling-prometheus-with-thanos-a-guide-to-long-term-scalable-monitoring-a1eca334cbd4
[45] How to Deploy a Scalable Prometheus with Thanos on K8s Using Terraform - DevOps.dev, accessed September 13, 2025, https://blog.devops.dev/how-to-deploy-a-scalable-prometheus-with-thanos-on-k8s-using-terraform-05a2626edd60
[46] Thanos - Highly available Prometheus setup with long term storage capabilities, accessed September 13, 2025, https://thanos.io/
[47] Thanos vs Cortex | Last9, accessed September 13, 2025, https://last9.io/blog/thanos-vs-cortex/
[48] Prometheus Cortex - OpsRamp, accessed September 13, 2025, https://www.opsramp.com/guides/prometheus-monitoring/prometheus-cortex/
[49] Scaling Prometheus: From Single Node to Enterprise-Grade Observability - Oodle AI, accessed September 13, 2025, https://blog.oodle.ai/scaling-prometheus-from-single-node-to-enterprise-grade-observability/
[50] ELK Stack Made Simple: Logs, Analytics & Visualizations Explained - Talent500, accessed September 13, 2025, https://talent500.com/blog/what-is-elk-stack/
[51] Log Management With ELK and Why You Should

Care - Cprime, accessed September 13, 2025, https://www.cprime.com/resources/blog/log-management-elk-and-why-you-should-care/
[52] Elastic Stack: (ELK) Elasticsearch, Kibana & Logstash, accessed September 13, 2025, https://www.elastic.co/elastic-stack
[53] Deep Dive into Elastic Stack (ELK): Elasticsearch, Logstash, and Kibana | by Amanat Ansari, accessed September 13, 2025, https://medium.com/@amanatansari07/deep-dive-into-elastic-stack-elk-elasticsearch-logstash-and-kibana-d8a0eb182cdb
[54] A Deep Dive into Log Monitoring Using Elastic Stack - QBurst Blog, accessed September 13, 2025, https://blog.qburst.com/2020/01/a-deep-dive-into-log-monitoring-using-elastic-stack/
[55] Kibana Tutorial 2025: Elasticsearch Visualization Made Simple - Knowi, accessed September 13, 2025, https://www.knowi.com/blog/kibana-an-overview-of-the-data-visualization-tool/
[56] Elastic Beats & Where They Fit With ELK Stack | Instaclustr, accessed September 13, 2025, https://www.instaclustr.com/blog/elastic-beats-and-where-they-fit-with-elk-stack/
[57] Centralised Logging System Using ELK and Filebeat | by Shreetheja S N - Medium, accessed September 13, 2025, https://medium.com/@snshagri/centralised-logging-system-using-elk-and-filebeat-067e1373dd71
[58] Difference Between Beats and Elastic Agent - DevOpsSchool.com, accessed September 13, 2025, https://www.devopsschool.com/blog/difference-between-beats-and-elastic-agent/
[59] What are Beats, how do they work. What are Beats? | by Cyber Tool Guardian | Medium, accessed September 13, 2025, https://medium.com/@cybertoolguardian/what-are-beats-how-do-they-work-adc21f209b3
[60] Configuring Logstash Pipeline for Data Processing - GeeksforGeeks, accessed September 13, 2025, https://www.geeksforgeeks.org/elasticsearch/configuring-logstash-pipeline-for-data-processing/
[61] How Logstash Works - Elastic, accessed September 13, 2025, https://www.elastic.co/docs/reference/logstash/how-logstash-works
[62] Getting Started with Logstash - Your First Steps in Data Processing - MoldStud, accessed September 13, 2025, https://moldstud.com/articles/p-getting-started-with-logstash-your-first-steps-in-data-processing
[63] Creating a Logstash Pipeline - Elastic, accessed September 13, 2025, https://www.elastic.co/docs/reference/logstash/creating-logstash-pipeline
[64] 7 Ways to Optimize Your Elastic (ELK) Stack in Production | Better Stack Community, accessed September 13, 2025, https://betterstack.com/community/guides/scaling-elastic-stack/optimize-elastic-stack/
[65] Optimizing Your ELK Stack: 7 Ways To Better Production | Opstergo Blog, accessed September 13, 2025, https://www.opstergo.com/blog/optimizing-elk-stack-better-production
[66] Elasticsearch Architecture: 7 Key Components | NetApp, accessed September 13, 2025, https://www.netapp.com/learn/cvo-blg-elasticsearch-architecture-7-key-components/
[67] What is Elasticsearch? Complete Guide for 2025 (How It Works) - Knowi, accessed September 13, 2025, https://www.knowi.com/blog/what-is-elastic-search/
[68] Prometheus vs. ELK - MetricFire, accessed September 13, 2025, https://www.metricfire.com/blog/prometheus-vs-elk/
[69] ElasticSearch Architecture: A Comprehensive Guide - DEV Community, accessed September 13, 2025, https://dev.to/wadee_sami_4562c11ecf8066/elasticsearch-architecture-a-comprehensive-guide-12me
[70] The Ultimate Guide to ELK Log Analysis - ChaosSearch, accessed September 13, 2025, https://www.chaossearch.io/blog/ultimate-guide-elk-log-analysis
[71] Scaling Elasticsearch by Cleaning the Cluster State - Sematext, accessed September 13, 2025, https://sematext.com/elasticsearch-scaling-cluster-state/
[72] Loki vs Elasticsearch - Which tool to choose for Log Analytics? - SigNoz, accessed September 13, 2025, https://signoz.io/blog/loki-vs-elasticsearch/
[73] Comparing ELK, Grafana, and Prometheus for Observability - Last9, accessed September 13, 2025, https://last9.io/blog/elk-vs-grafana-vs-prometheus/
[74] How to Create a Dashboard in Kibana - ChaosSearch, accessed September 13, 2025, https://www.chaossearch.io/blog/how-to-create-kibana-dashboard
[75] Explore and analyze data with Kibana | Elastic Docs, accessed September 13, 2025, https://www.elastic.co/docs/explore-analyze
[76] Prometheus vs Elasticsearch stack - Key concepts, features, and differences - SigNoz, accessed September 13, 2025, https://signoz.io/blog/prometheus-vs-elasticsearch/
[77] Maximizing Elasticsearch performance when adding nodes to a cluster | Elastic Blog, accessed September 13, 2025, https://www.elastic.co/blog/maximizing-elasticsearch-performance-when-adding-nodes-to-a-cluster
[78] Elasticsearch scaling considerations | Elastic Docs, accessed September 13, 2025, https://www.elastic.co/docs/deploy-manage/production-guidance/scaling-considerations
[79] Scaling with Elasticsearch: use cases - Severalnines, accessed September 13, 2025, https://severalnines.com/blog/scaling-with-elasticsearch-use-cases/
[80] Kubernetes Monitoring with Grafana, accessed September 13, 2025, https://grafana.com/solutions/kubernetes/
[81] Grafana Guide | InfluxData, accessed September 13, 2025, https://www.influxdata.com/grafana/
[82] Graphite and Grafana | MetricFire, accessed September 13, 2025, https://www.metricfire.com/blog/graphite-and-grafana/
[83] Visualization and monitoring solutions | Grafana

Labs, accessed September 13, 2025, https://grafana.com/solutions/
[84] Graphite OSS | Time-series data platform - Grafana, accessed September 13, 2025, https://grafana.com/oss/graphite/
[85] Grafana - How to read Graphite Metrics - MetricFire, accessed September 13, 2025, https://www.metricfire.com/blog/grafana-how-to-read-graphite-metrics/
[86] Kubernetes Monitoring Stack | DigitalOcean Marketplace 1-Click App, accessed September 13, 2025, https://marketplace.digitalocean.com/apps/kubernetes-monitoring-stack
[87] InfluxDB monitoring made easy | Grafana Labs, accessed September 13, 2025, https://grafana.com/solutions/influxdb/monitor/
[88] InfluxDB Overview | Grafana Labs, accessed September 13, 2025, https://grafana.com/grafana/dashboards/13109-influxdb-oss-overview/
[89] InfluxDB OSS Stats monitoring dashboard | Grafana Labs, accessed September 13, 2025, https://grafana.com/grafana/dashboards/10346-influxdb-oss-stats-monitoring-dashboard/
[90] Grafana Cloud Graphite Monitoring Tools, accessed September 13, 2025, https://grafana.com/go/hosted-graphite-monitoring/
[91] Loki vs. Elasticsearch: Choosing the Right Logging System for You - KubeBlogs, accessed September 13, 2025, https://www.kubeblogs.com/loki-vs-elasticsearch/
[92] Grafana Loki vs ELK Logging Stacks - Wallarm, accessed September 13, 2025, https://www.wallarm.com/cloud-native-products-101/grafana-loki-vs-elk-logging-stacks
[93] Grafana Loki — Our journey on replacing Elastic Search and adopting a new logging solution at Arquivei | by João Guilherme Luchetti | Engenharia Qive | Medium, accessed September 13, 2025, https://medium.com/engenharia-arquivei/grafana-loki-our-journey-on-replacing-elastic-search-and-adopting-a-new-logging-solution-at-f65aec407e47
[94] Guide - Custom vs Off-the-Shelf Software: Pros and Cons - Scrums.com, accessed September 13, 2025, https://www.scrums.com/guides/the-pros-and-cons-of-custom-vs-off-the-shelf-software-development
[95] Custom Software vs Off-the-Shelf: Hidden Costs & Benefits Revealed - Netguru, accessed September 13, 2025, https://www.netguru.com/blog/custom-software-vs-off-the-shelf
[96] I Built My Company's First Monitoring System — Here's What I Learned | by Naomi Kriger | Data Science Collective - Medium, accessed September 13, 2025, https://medium.com/data-science-collective/i-built-my-companys-first-monitoring-system-here-s-what-i-learned-bed76942cc72
[97] Custom Software vs. Off-the-Shelf Solutions: A Complete Cost-Benefit Analysis for Growing Businesses - Full Scale, accessed September 13, 2025, https://fullscale.io/blog/custom-software-vs-off-the-shelf-cost-analysis/
[98] Custom vs. Off-the-Shelf: Software Solutions for You From GovPilot, accessed September 13, 2025, https://www.govpilot.com/blog/custom-vs.-off-the-shelf-software-solutions-for-you-from-govpilot
[99] How to Build a Data Monitoring System (Quickest Way to Start) - Telmai, accessed September 13, 2025, https://www.telm.ai/blog/how-to-build-a-data-monitoring-system/
[100] Create and manage custom dashboards - Monitoring - Google Cloud, accessed September 13, 2025, https://cloud.google.com/monitoring/charts/dashboards
[101] Getting Started with Dashboards - Datadog Docs, accessed September 13, 2025, https://docs.datadoghq.com/getting_started/dashboards/
[102] Prometheus vs. Elasticsearch - Atatus, accessed September 13, 2025, https://www.atatus.com/blog/prometheus-vs-elasticsearch/
[103] Prometheus vs ELK Stack: Choosing the Right Monitoring Solution for Your Needs, accessed September 13, 2025, https://www.tetrain.com/blogs/post/117/prometheus-vs-elk-stack-choosing-the-right-monitoring-solution-for-your-needs.html
[104] Comparison of Time-Series Databases: InfluxDB vs. Prometheus - GeeksforGeeks, accessed September 13, 2025, https://www.geeksforgeeks.org/blogs/influxdb-vs-prometheus/
[105] Prometheus vs InfluxDB [Detailed Technical Comparison for 2025], accessed September 13, 2025, https://uptrace.dev/comparisons/prometheus-vs-influxdb
[106] Uber | CNCF, accessed September 13, 2025, https://www.cncf.io/case-studies/uber/
[107] M3: Uber's Open Source, Large-scale Metrics Platform for Prometheus | Uber Blog, accessed September 13, 2025, https://www.uber.com/blog/m3/
[108] arRESTful Development: How Netflix Uses Elasticsearch to Better ..., accessed September 13, 2025, https://www.elastic.co/elasticon/conf/2015/sf/arrestful-development-how-netflix-uses-elasticsearch-to-better-understand
[109] Solved: Prometheus vs ELK Stack - Which is better for Monitoring? - Squadcast, accessed September 13, 2025, https://www.squadcast.com/compare/prometheus-vs-elk-stack-a-comprehensive-comparison-of-monitoring-and-logging-solutions
[110] Prometheus user profile: How DigitalOcean uses Prometheus | CNCF, accessed September 13, 2025, https://www.cncf.io/blog/2017/02/28/prometheus-user-profile-digitalocean-uses-prometheus/
[111] Centralized Logging with Elastic Search, Logstash & Kibana — ELK Stack - Medium, accessed September 13, 2025, https://medium.com/@dineshmurali/centralized-logging-with-elastic-search-logstash-kibana-elk-stack-81e7ff2b0d34
[112] What Is OpenTelemetry? A Complete Guide | Splunk, accessed September 13, 2025, https://www.splunk.com/en_us/blog/learn/opentelem

etry.html
[113] What is OpenTelemetry? An open-source standard for logs, metrics, and traces - Dynatrace, accessed September 13, 2025, https://www.dynatrace.com/news/blog/what-is-opentelemetry/
[114] CNCF Annual Survey 2023, accessed September 13, 2025, https://www.cncf.io/reports/cncf-annual-survey-2023/
[115] CNCF Annual Survey 2022, accessed September 13, 2025, https://www.cncf.io/reports/cncf-annual-survey-2022/