sysdig

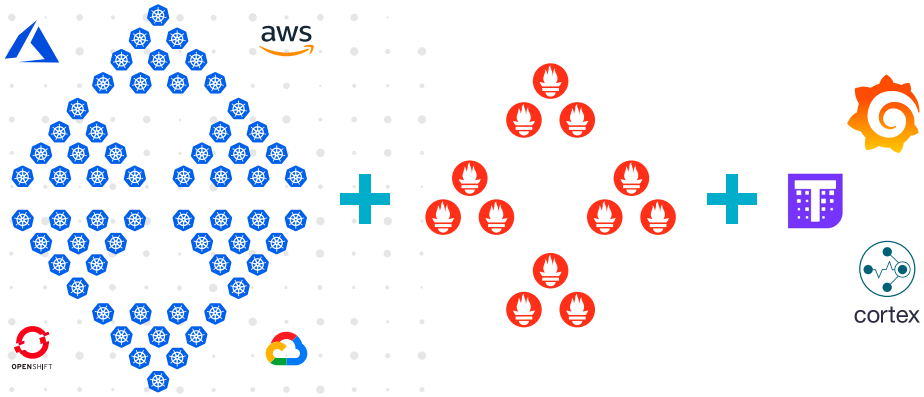
Six Keys for Scaling Prometheus

Containers are ephemeral and difficult to troubleshoot. They require comprehensive, granular and context-rich data to anticipate and prevent issues that can impact user experience. In order to effectively manage the cloud native applications, DevOps teams require a toolset designed specifically to analyze, predict and prevent issues in containers, Kubernetes services, and Cloud-native infrastructure.

Prometheus has become the de-facto standard for DevOps / SRE teams to monitor Kubernetes workloads. It offers a number of advantages - it's easy to set up, implement and maintain on a single server. Prometheus is also well suited for highly dynamic Kubernetes-based environments.

As the scope of Kubernetes development and production deployment increases in enterprises, DevOps / SRE teams quickly find the need to scale Prometheus-based monitoring capability. Several open source tools are available for scaling Prometheus.

Figure 1: Do-It-Yourself Scaled Prometheus Technology Reference Architecture

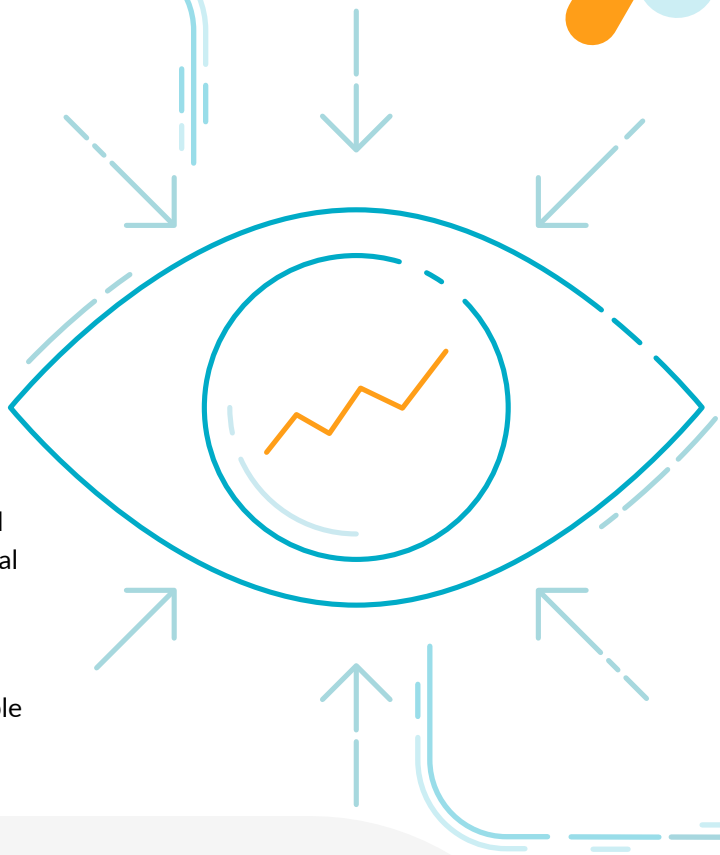


A technology reference architecture is outlined above for reference. Here are the six keys for scaling Prometheus:

1

6 KEYS FOR SCALING PROMETHEUS

Centralize Visualization & Analytics Capability



Often, each DevOps team spins up their own instances of Kubernetes, Prometheus and Grafana in multiple environment configurations (dev, test, prod) to monitor workloads independently. Enterprises end up with multiple visualization and analytics tool instances, such as Grafana. However, a lack of global view across multiple Kubernetes clusters and cloud stacks poses problems like:

- RBAC and security policies need to be duplicated across multiple instances.
- Developers will take more time to diagnose complex problems.

Consolidate Grafana Instances. The first step is to consolidate Grafana instances by migrating and consolidating configurations, dashboards and alerts. The global view helps DevOps teams monitor the entire Kubernetes install base in a consistent approach, and enables the teams to conduct faster root cause analysis.

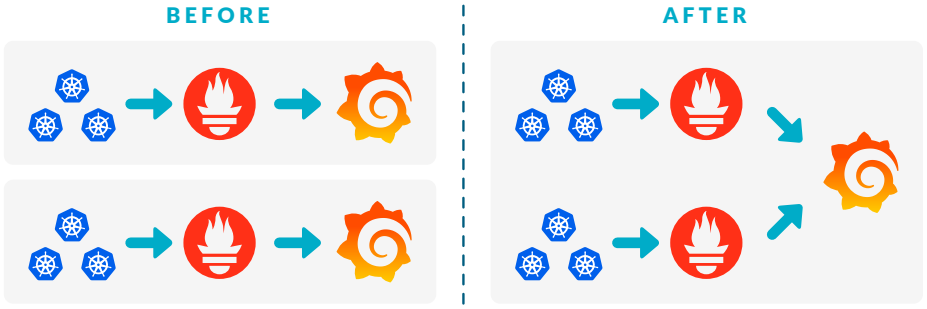
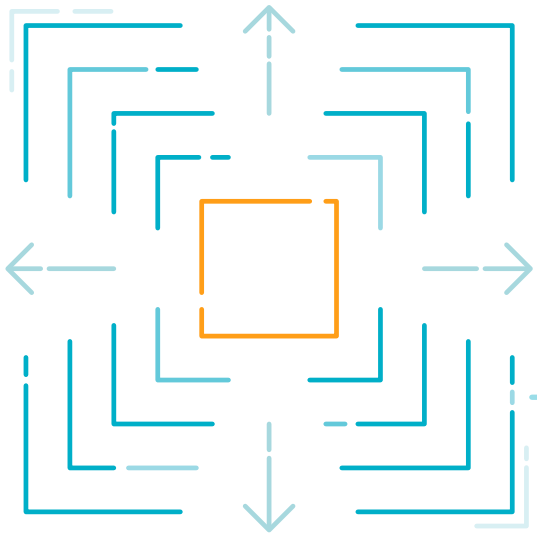


Figure 2: Centralize Visualization & Analytics Capability

2

6 KEYS FOR SCALING PROMETHEUS

Scale Vertically and then Horizontally



A single Prometheus server can easily handle millions of time series. As your systems scale beyond that, Prometheus can scale too. There are a few options for scaling Prometheus - vertical, hierarchial federation as well as horizontal sharding. Each approach has certain limitations and we have outlined them below.

While vertically scaling Prometheus is a good starting point, this approach has limitations. As DevOps reach the limits of vertical scaling, they experience availability issues and thus a loss of metrics.

Functional sharding offers a highly scalable Prometheus solution. Horizontal sharding approach enables splitting monitoring jobs among many Prometheus nodes, while still ensuring a consistent top-level view of metrics for graphing and alerting purposes. Horizontal sharding can be achieved by splitting jobs by namespace, geo, customer, environment type. Uneven and volatile monitoring data makes it difficult to develop correct granularity for functional sharding. This leads to reduced levels of granularity of metrics data at global view, as well as higher hardware and operational Costs.

This step is more complex to setup and requires much more involvement to manage than a normal Prometheus deployment, so it should be avoided for as long as possible. It is imperative that the DevOps teams build a sandbox to test in before releasing functional sharding capability to Production.

Hierarchical federation of Prometheus servers may be an alternative for some use cases. Hierarchical federation allows Prometheus to scale to environments with tens of data centers and millions of nodes. In this use case, the federation topology resembles a tree, with higher-level Prometheus servers collecting aggregated time series data from a larger number of subordinated servers. Hierarchical federation of Prometheus can be used to address business continuity requirements – global monitoring still works even when some nodes experience outage. However, hierarchical federation leads to reduced levels of granularity of metrics data at global view.

3

6 KEYS FOR SCALING PROMETHEUS

Enable Long-term Retention and Down Sampling

Cloud native applications generate High Dimensionality (HD) metric data. In HD or High Cardinality metric data, each metric can be broken down to several sub components or metrics. Querying and storing metrics with all of the dimensions quickly becomes cost prohibitive, slow, and impossible to use.

Prometheus, by design, is not built for long-term storage of metrics as it uses only one datastore. Lack of long-term storage leads to performance issues due to insufficient visibility. Cloud-scale system implementations can produce metrics data in the order of terabytes per day. In order to address the multitude of storage requirements, DevOps teams need to augment Prometheus with a long-term storage solution.

Extend Prometheus deployment to include long-term storage. There are open source solutions, including Thanos or Cortex, and a supported commercial solution by Sysdig. DevOps teams need to assess the deployment options, evaluate the tradeoffs and implement the solution that best fits their needs.

Retain data to enable trend and comparative analysis. Historical data can be used to conduct Service Level Objectives (SLO) trend analysis, to compare SLOs across multiple workloads, geos and namespaces, and to potentially implement corrective actions to improve SLOs.

Implement policies to improve corporate compliance. Enterprises tend to have specific data retention policies. These policies govern the type of metrics that need to be captured, the retention period, and the type of storage required (operational data store vs. backup datastore).

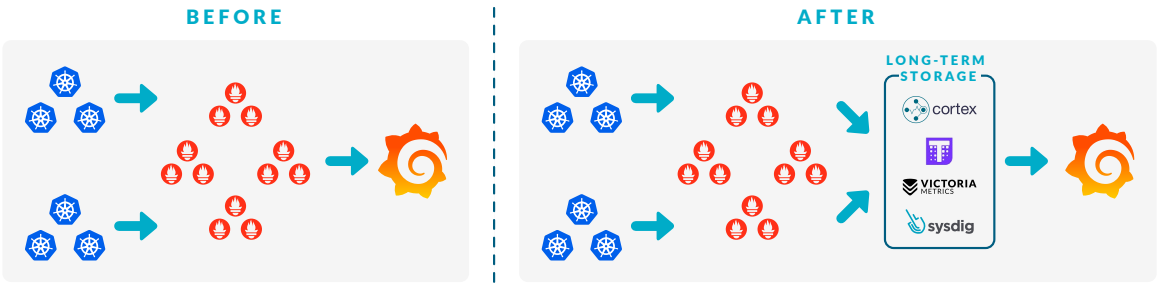


Figure 3: Enable Long-term Storage, Down sampling and Variable Metric Retention

4

Improve Corporate Compliance with Robust Access Controls

Mission critical systems need to adhere to corporate and regulatory compliance requirements. Security is a critical requirement for any enterprise-grade deployment. Any tool or software should support two capabilities – encrypted communication and access control. TLS (Transport Layer Security) encryption of the HTTP endpoints and Role-Based Access Control (RBAC) are not available in out-of-the-box Prometheus, Grafana and Cortex distributions.

Use reverse proxy and apply TLS at the network layer. Reverse proxies are typically implemented to help increase security, performance, and reliability. Install reverse proxy in front of Prometheus servers to apply encryption at proxy layer.

Use security features in Grafana and Cortex to improve security posture. RBAC functionality is not available out-of-box. Grafana enables configuration of users and groups to limit access to predefined dashboards. Cortex enables multi-tenancy to restrict access at a tenant level. Alternatively, DevOps teams can integrate SSO through an authentication proxy to provide more granular levels of security. RBAC can be applied at storage, visualization, and analytics layers by integrating them into enterprise-owned Single Sign-On (SSO) software.

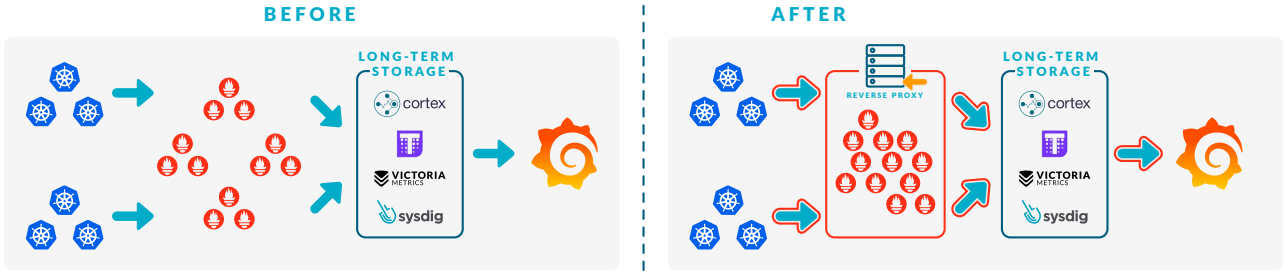
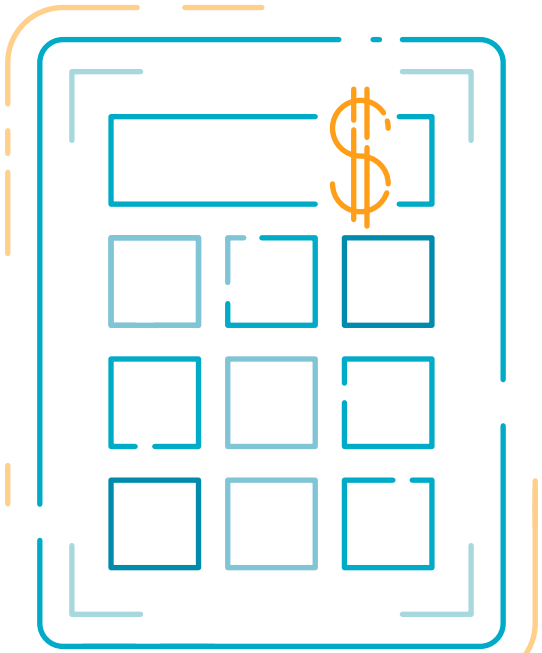


Figure 4: Improve Corporate Compliance with Role-based Access Control (RBAC)

5

6 KEYS FOR SCALING PROMETHEUS

Track and allocate monitoring costs



As DevOps teams expand their usage of Prometheus, they are looking for ways to use the resources in efficient ways by sharing resources across multiple teams (groups) and users. The operations teams are looking for a mechanism to distribute and allocate costs associated with operating these platforms across the teams by 'charging them back' for their use.

Develop chargeback to the customers and enable them to optimize expenses. The first step is to collect the information and present it to the DevOps teams (i.e., 'show back'). Breaking it down further, there are three steps for operation teams to enable chargeback:

- Understand consumption by collecting the desired metrics at the desired granularity (i.e., usage).
- Co-relate consumption to the selected grouping using labels, tags, etc. (i.e., show back).
- Integrate show back into internal reporting and billing systems for budget-based IT cost allocation (i.e., chargeback).

You must also consider the cost of managing all your Prometheus deployments over time as it can scale up very quickly. Application exporters and yaml configurations must be constantly evaluated and kept up to date. Every full-time equivalent resource that is dedicated to maintaining DIY Prometheus at scale is one less developer or SRE that could be writing innovative code that transforms and grows your business.

6

6 KEYS FOR SCALING PROMETHEUS

DIY Build vs. Buy

Prometheus has been a game-changer in the monitoring landscape for cloud-native applications, just like Kubernetes has been to container orchestration. However, even large tech companies find the challenges of managing and scaling Prometheus in a DIY fashion daunting. With so many yaml configurations and the complicated exporter landscape to maintain, configuration errors and incompatibilities will eventually creep into your Promentheus environments.

To simplify this complexity, organizations should consider a managed Prometheus solution that provides out-of-the-box functionality to enable Prometheus monitoring and scaling without sacrificing open source compatibility. Let's consider Sysdig's managed Prometheus service as an example:

DIY vs. Sysdig

DIY	Sysdig
PromQL only for queries	**PromQL and/or simple Form UI to build queries**
Imported community dashboards	**Automatic pre-configured dashboards**
Lots of time managing exporters	**Automated discovery and assisted configuration of curated exporters**
Several weeks of time series storage	**Long term storage built-in**
Configure and manage Thanos or Cortex	**Built-in scaling with no management required**
DIY Access Controls	**Support for RBAC, SSO, LDAP, and Sysdig Teams**

Conclusion

Organizations should consider these six areas when considering how they will scale Prometheus:

- Centralization of metrics
- Vertical vs. horizontal scaling
- Long term metric retention
- Compliance and access controls
- Tracking costs
- Managed service options

Sysdig offers an enterprise managed Prometheus service that can help avoid the cost and complexity associated with scaling and running multiple Prometheus deployments. Visit sysdig.com or contact us to learn how easy Prometheus monitoring can be!