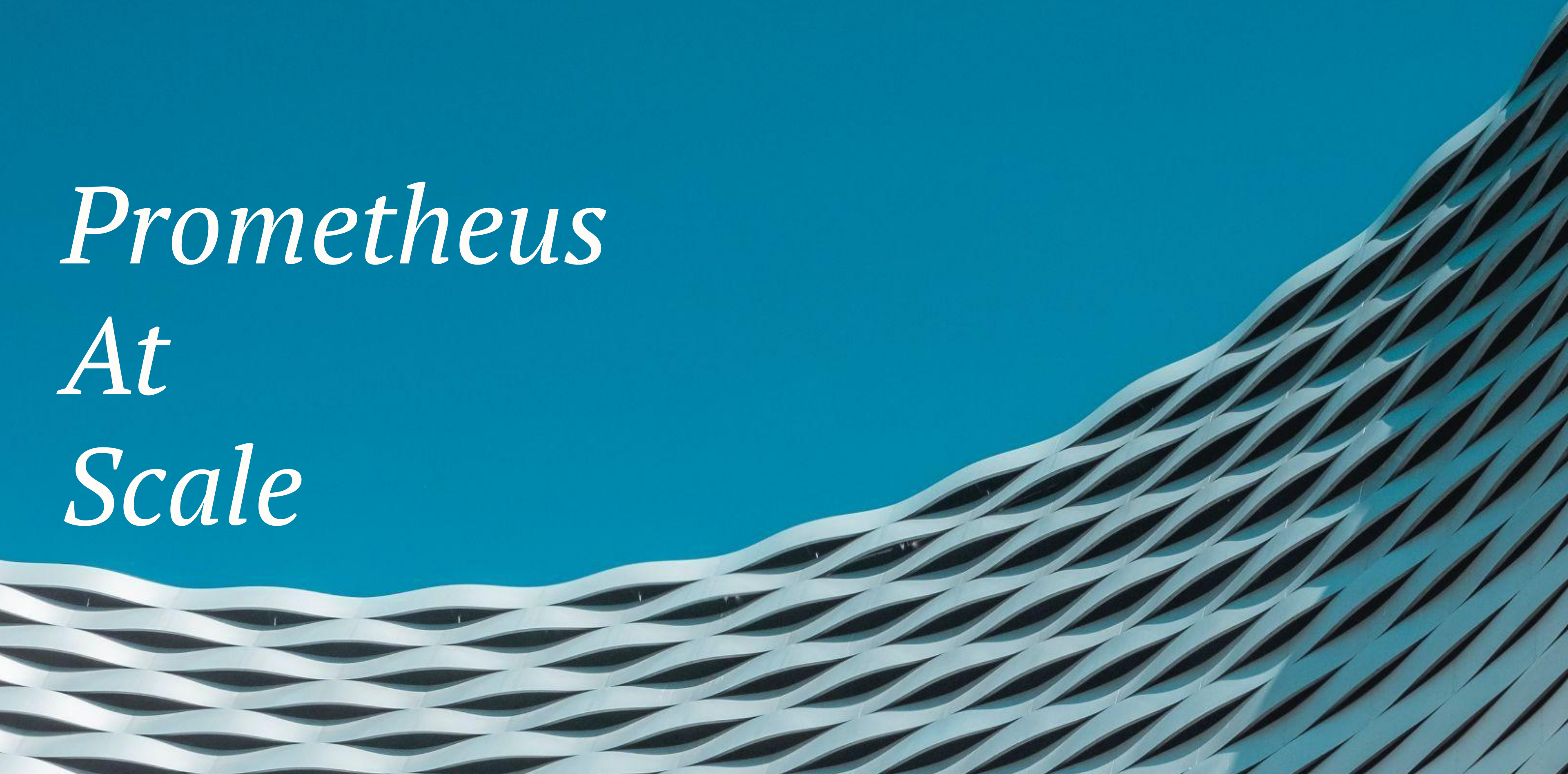
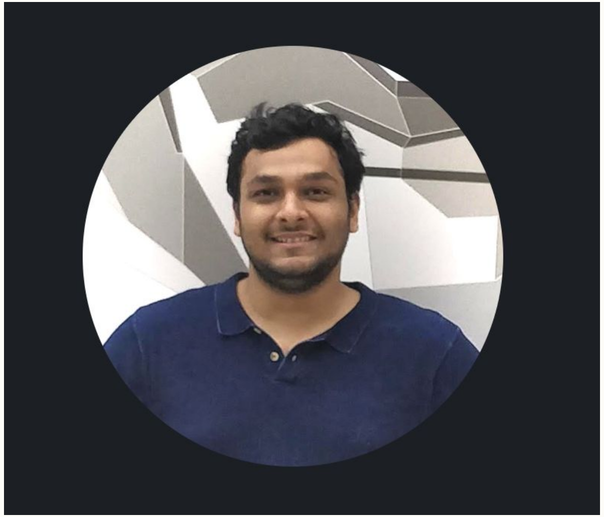
Prometheus At Scale



Introduction



Hi, I am **Abhishek Garg**. Currently working as *DoE* in **Syfe**.

Today I will be talking about one of architecture project we did back in 2018, when prometheus was getting pace and Thanos was very new.

I will talking about how we did migrate our observability from a enterprise to a in-house monitoring system which eventually helps us save millions of USD / Year.

Let’s talk about Scale first !!

Nodes / Endpoints	Data Ingestion sum(scrape_samples_scraped)	Retention	Cost (DataDog)
25k	**300 Million / minute**	**30-60 Days**	**$ 6 Million / year ***

***Datadog Current Pricing**

- ***APM*** ➔ *$31 – $45 (range reflects different tiers and access to features) / host / month*
- ***Infrastructure Monitoring*** ➔ *$15 – $34 (range reflects different tiers and access to features) /host /month*

*Current Base price around **USD 14 million / Year***

What is Prometheus ??

What it is for

Metrics-based monitoring & alerting stack.

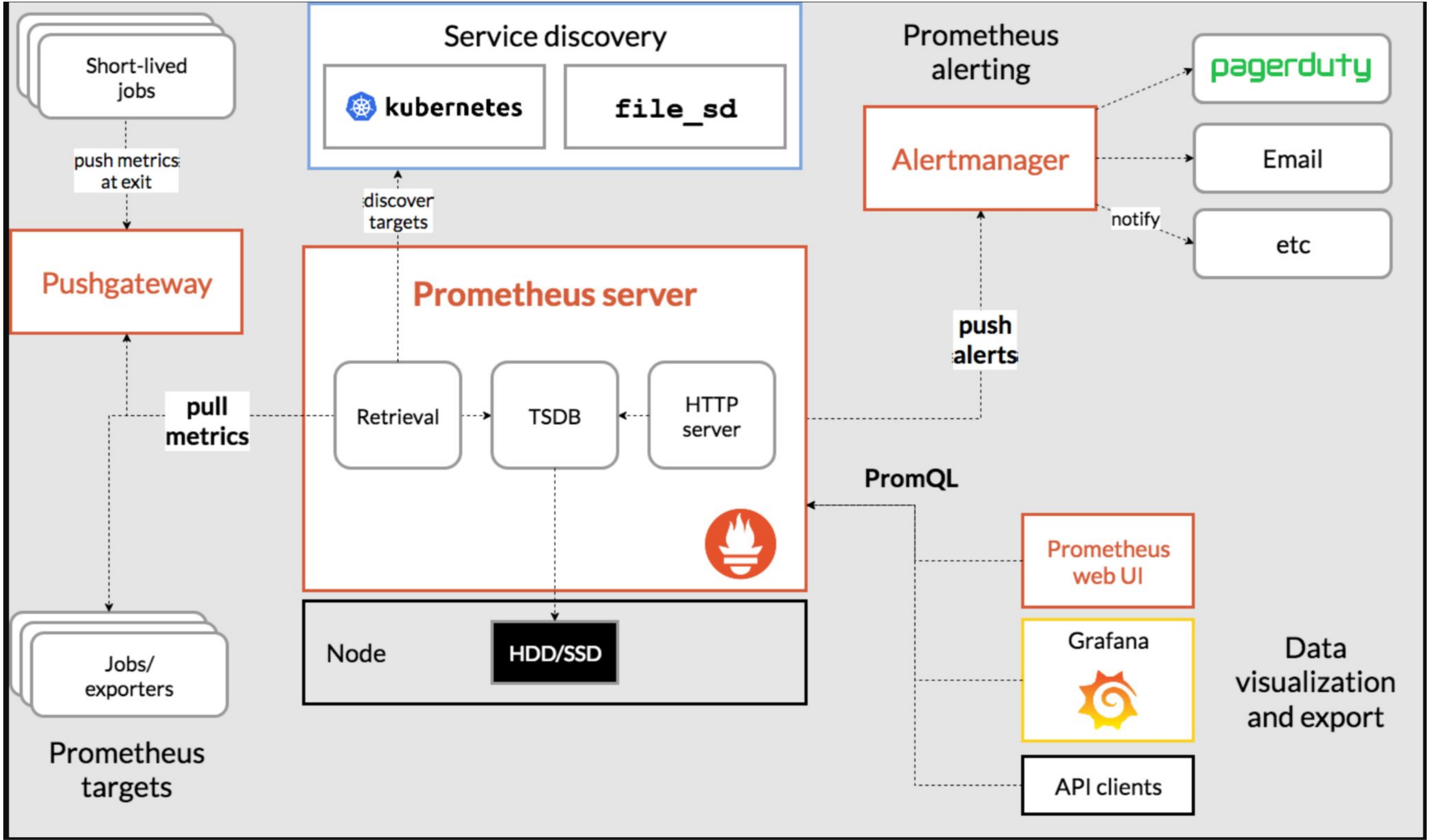
- Instrumentation
- Metrics collection and storage
- Querying, alerting, dashboarding
- For all levels of the stack!

What it is not for

Logging, Tracing and ….

- Logging, Event Collection or tracing
- Automatic anomaly detection
- Scalable or durable storage
- Automatic Horizontal scaling
- User authorization management

Basic Architecture Of Prometheus


Short-lived jobs
push metrics at exit
Pushgateway
Service discovery
kubernetes
file_sd
discover targets
Prometheus alerting
pagerduty
Alertmanager
Email
notify
etc
Prometheus server
Retrieval
TSDB
HTTP server
pull metrics
push alerts
PromQL
Prometheus web UI
Grafana
API clients
Data visualization and export
Node
HDD/SSD
Jobs/ exporters
Prometheus targets

It’s Simple, right?

Not Much, here are some problems!

1. **Scalability Issue**
    a. *Single Server Limitations:* Prometheus is designed as a single-node system, which can become a bottleneck when dealing with high cardinality metrics and large-scale environments.
    b. *Storage Limitations:* The local storage of Prometheus can be a limiting factor for long-term storage and high-frequency data points.

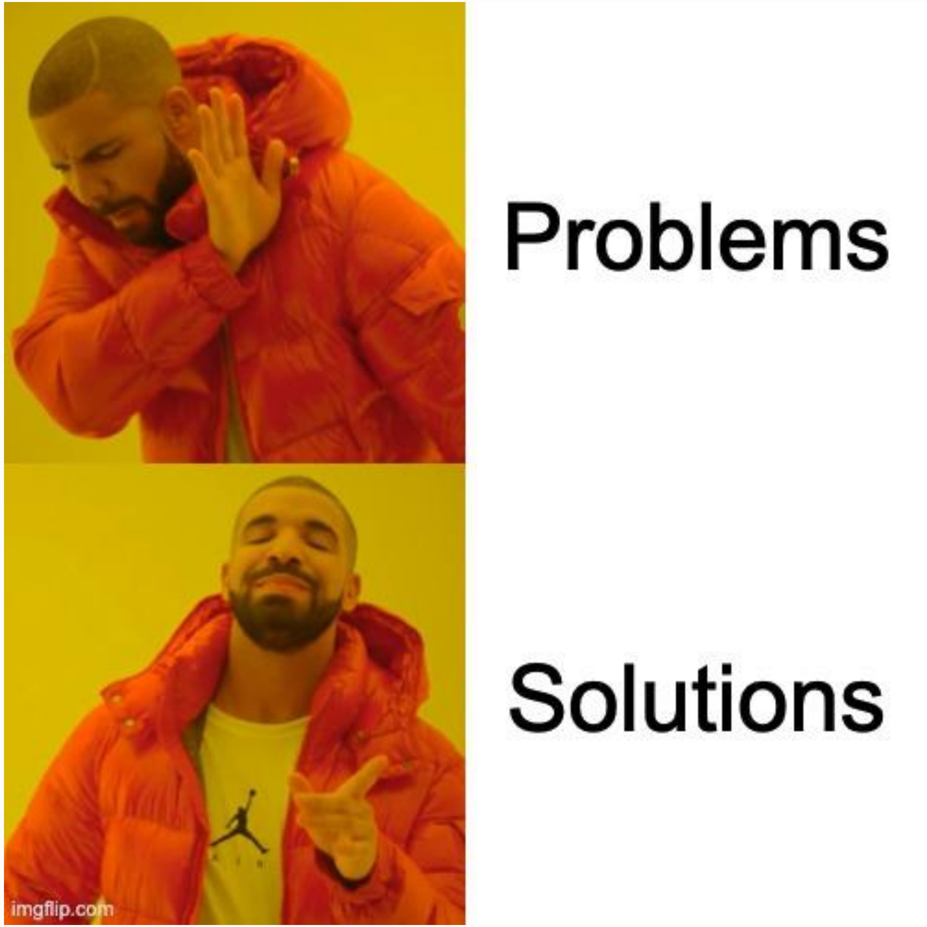
2. **High Cardinality and Performance**
    b. *High Cardinality Metrics:* Prometheus can struggle with high cardinality metrics (metrics with a large number of unique label combinations), which can lead to increased memory usage and degraded performance.
    c. *Query Performance:* Complex queries over large datasets can be slow and resource-intensive, impacting the responsiveness of the system.

more..

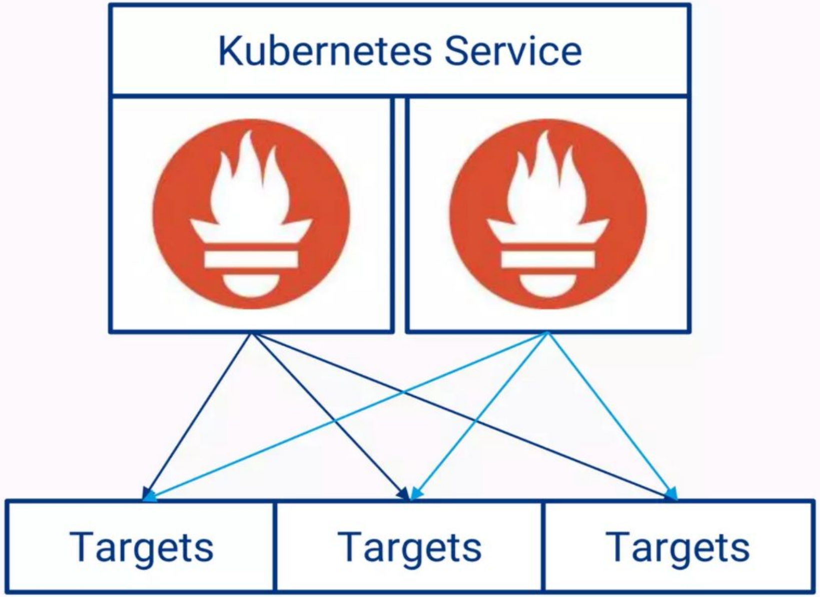
3. **Configuration Management**
    a. *Static Configuration***:** Prometheus relies on static configuration files for setting up scrape targets and rules, which can be cumbersome to manage in dynamic and large-scale environments.

4. **Operational Overhead**
    b. *Maintenance and Upgrades:* Maintaining Prometheus, including handling updates, scaling, and troubleshooting, can require significant operational effort and expertise.

Let’s talk business!



Highly Available Prometheus!!!



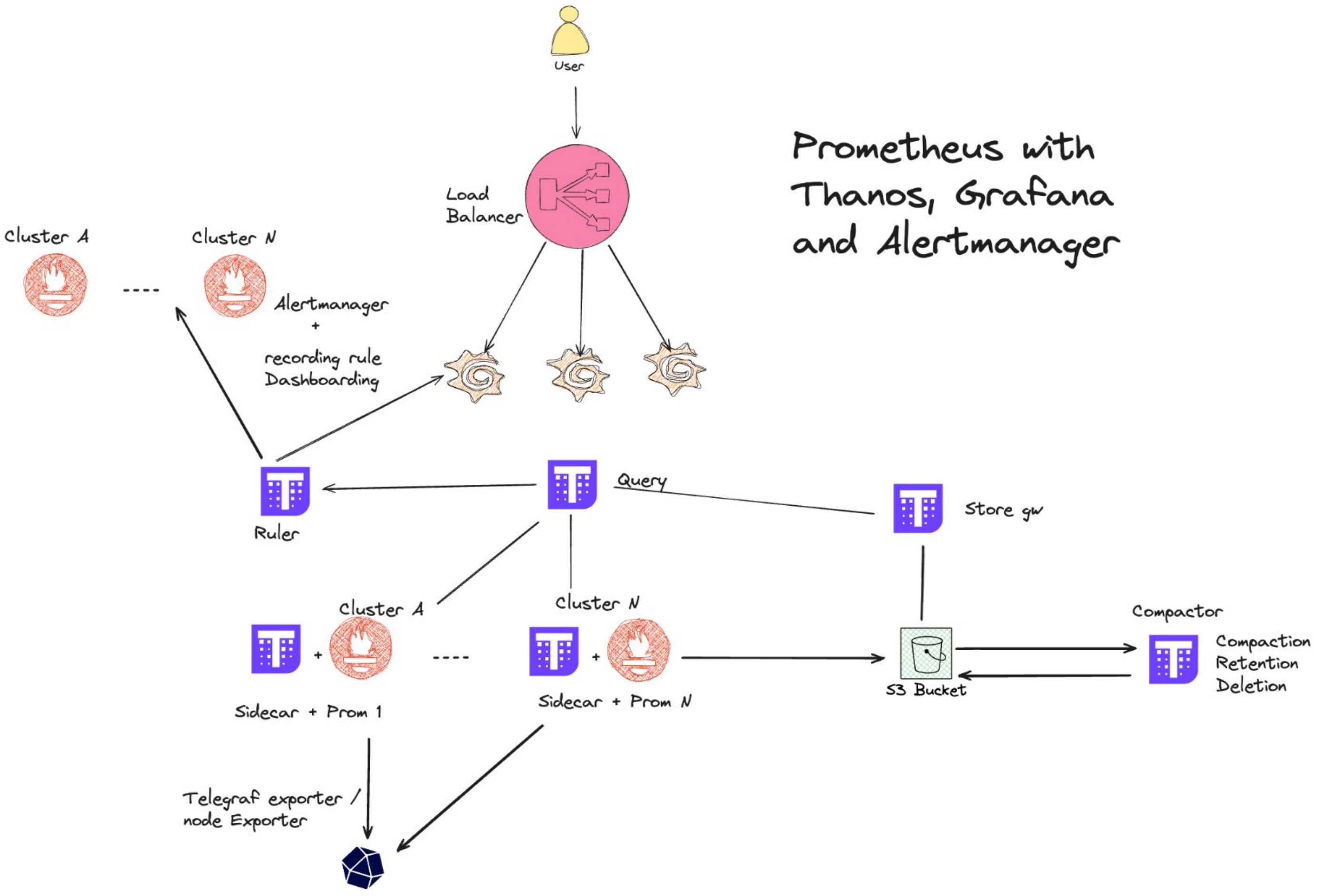
Not without its challenges:

- When you refresh the data, you will see it change as metrics will potentially differ between the two instances

Thanos

Thanos is a set of components that can be composed into a highly available metric system with unlimited storage capacity, which can be added seamlessly on top of existing Prometheus deployments.

Prometheus with Thanos, Grafana and Alertmanager



Seems Costly ?

.

.

.

.

.

.

.

.

Let’s Calculate!

Components Used	Thanos, Prometheus, Grafana, Object Storage, ELB
RAM requirement for ***300 million / min*** metrics	Approx 7000 GB
Storage requirements for above estimates metrics	46 TB (46000 GB)
Cost of Graviton Instance (1 Year Upfront) / GB RAM / year	36.2 USD
S3 Cost / GB	0.08 (assuming 10000 RW call / GB)

Compute total Cost = (7000 * 36.2) = 253400 USD

Storage total Cost = (46000 * 0.08 * 12) = 44160 USD

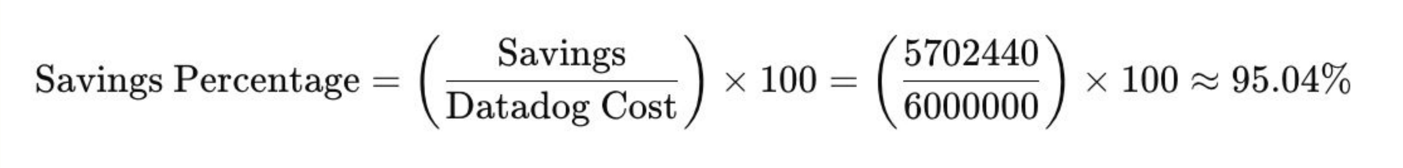
Final Cost = Compute + Storage = (253400 + 44160) = ***2,97,560 USD / Year***

Savings:

Savings = Datadog Cost − Total Cost for Prometheus + Thanos

= 6000000 − 297560 = 5702440 USD/year

Savings Percentage:



$$\text{Savings Percentage} = \left(\frac{\text{Savings}}{\text{Datadog Cost}}\right) \times 100 = \left(\frac{5702440}{6000000}\right) \times 100 \approx 95.04\%$$

Summary:

- Savings Percentage: Approximately ***95.04%***

Note: *even with a* ***45% variance*** *added as noise to the cost of Prometheus + Thanos, you would still save approximately* ***92.81%*** *compared to using Datadog.*

Other Important points:

- ***Avoid Unnecessary Labels and metrics:*** Minimize the use of labels that can have a large number of unique values, such as timestamps, user IDs, session IDs, or request IDs. If possible try to combine them. Also drop excessive metrics or limit metric collection / host.
- ***Limit Label Values:*** Ensure that labels have a limited and predictable set of values. For example, instead of using exact URLs as labels, use URL patterns or endpoints.
- ***Aggregate Metrics:*** Use Prometheus recording rules to pre-aggregate metrics with high cardinality. This reduces the amount of data stored and queried. For example, aggregate metrics by time intervals or other meaningful dimensions.
- ***Downsampling:*** Apply downsampling techniques to reduce the resolution of historical data, retaining only the necessary level of detail.
- ***Optimized PromQL Queries:*** Write efficient PromQL queries that avoid scanning large datasets unnecessarily. Use functions like sum, avg, max, min, and rate to aggregate data effectively.
- ***Query Caching:*** Implement query caching mechanisms if possible to reduce the load on Prometheus when executing repeated queries.
- ***Monitoring and Alerting:*** Set up alerts to notify you when cardinality exceeds acceptable thresholds, enabling proactive management.

Meet the Team

Ram Shankar Jaiswal
Sr. Cloud Architect

Abhishek Garg
Director of Engineering

Q & A

Thanks

Scan here to connect me over LinkedIn