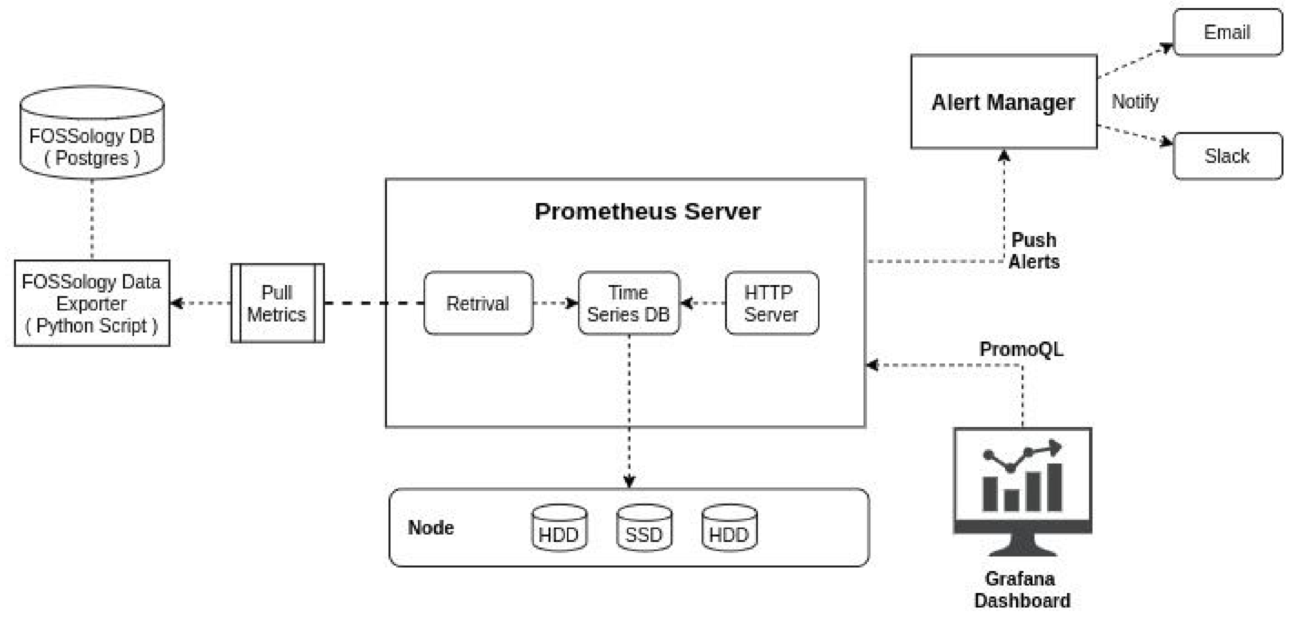
FOSSOlogy Dashboard Document

Abstract :

FOSSology generates a large set of data that can be exported to the time-series Prometheus database and visualized with the help of Grafana. I will write scripts that collect useful data from FOSSology DB (Postgres) and expose the into Prometheus DB. Develop a Monitoring dashboard in Grafana by integrating Prometheus as input data source. Also develop an alerting mechanism to notify admin via email.

Our Architecture :



Background Knowledge :

Prometheus :

Prometheus is an open-source software application used for event monitoring and alerting. It records real-time metrics in a time series database (allowing for high dimensionality) built using an HTTP pull model, with flexible queries and real-time alerting.

Prometheus works well for recording any purely numeric time series. It fits both machine-centric monitoring as well as monitoring of highly dynamic service-oriented architectures.

You can configure *prometheus.yml* file in the linux directory = */etc/prometheus/*
Prometheus provides 4 different types of metrics

1.Counter :

A *counter* is a cumulative metric that represents a single ***monotonically increasing counter*** whose value can only increase or be reset to zero on restart.

For example, you can use a counter to represent the number of requests served, tasks completed, or errors.

What are some use cases for counters?

- request count
- tasks completed
- error count

2.Gauge :

A *gauge* is a metric that represents a single numerical value that can arbitrarily **go up and down.**

Gauges are typically used for measured values like temperatures or current memory usage, resource utilization, but also "counts" that can go up and down, like the number of concurrent requests.

What are some use cases for gauges?

- memory usage
- queue size
- number of requests in progress

3.Histogram

A *histogram* samples observations (usual things like request durations or response sizes) and counts them in configurable buckets. It also provides a sum of all observed values.

When to use histograms?

- you want to take many measurements of value, to later calculate averages or percentiles

- you're not bothered about the exact values but are happy with an approximation

What are some use cases for histograms?

- request duration
- response size

Prometheus Terminology :

PromQL:

Prometheus provides its own query language PromQL (Prometheus Query Language) that lets users select and aggregate data. PromQL is specifically adjusted to work in convention with a Time-Series Database and therefore provides time-related query functionalities. Examples include the rate() function, the instant vector, and the range vector which can provide many samples for each queried time series.

Exporters:

There are a number of libraries and servers which help in exporting existing metrics from third-party systems as Prometheus metrics.

You can find a list of exporters at the following link:
https://prometheus.io/docs/instrumenting/exporters/

Alertmanager:

The Alertmanager handles alerts sent by client applications such as the Prometheus server.

Alerting with Prometheus is separated into two parts. Alerting rules in Prometheus servers send alerts to an Alertmanager. The Alertmanager then manages those alerts, including silencing, inhibition, aggregation, and sending out notifications via methods such as email, on-call notification systems, and chat platforms.

You can configure the *alertmanager.yml* file as per your need.
https://prometheus.io/docs/alerting/configuration/

Client libraries

Before you can monitor your services, you need to add instrumentation to their code via one of the Prometheus client libraries. These implement the Prometheus metric types.

Choose a Prometheus client library that matches the language in which your application is written. This lets you define and expose internal metrics via an HTTP endpoint on your application's instance:

Go, Java or Scala, Python, and Ruby

Grafana :

Grafana is open-source visualization and analytics software. It allows you to query, visualize, alert on, and explore your metrics no matter where they are stored.

The tool helps us study, analyze & monitor data over a period of time, technically called time-series analytics.

It helps us track the user behaviour, application behaviour, frequency of errors popping up in production or a pre-prod environment, type of errors popping up & the contextual scenarios by providing relative data.

There is built in the demo given by grafana Dashboard by integrating Prometheus as an input data source.
https://play.grafana.org/d/000000029/prometheus-demo-dashboard?orgId=1&refresh=5m

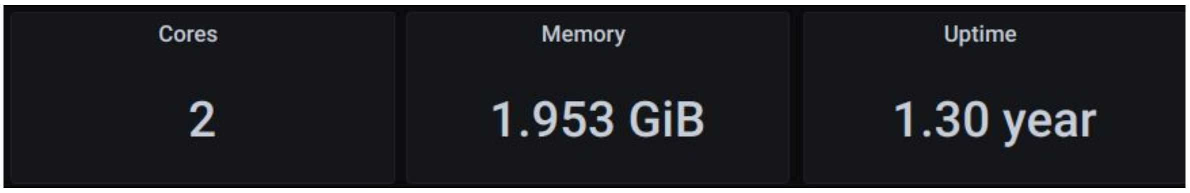
you can find out details of different visualization panel supported by grafana here
https://grafana.com/docs/grafana/latest/panels/visualizations/

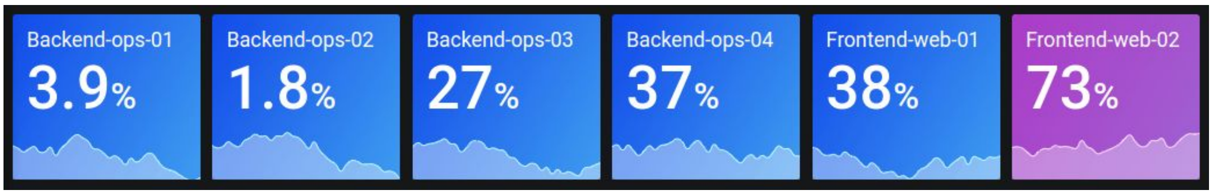
Some of the different visualizations supported by Grafana are as follow.

1.Simple stats

The Stat panel shows one large stat value with an optional graph sparkline. You can control the background or value color using thresholds.

Examples :

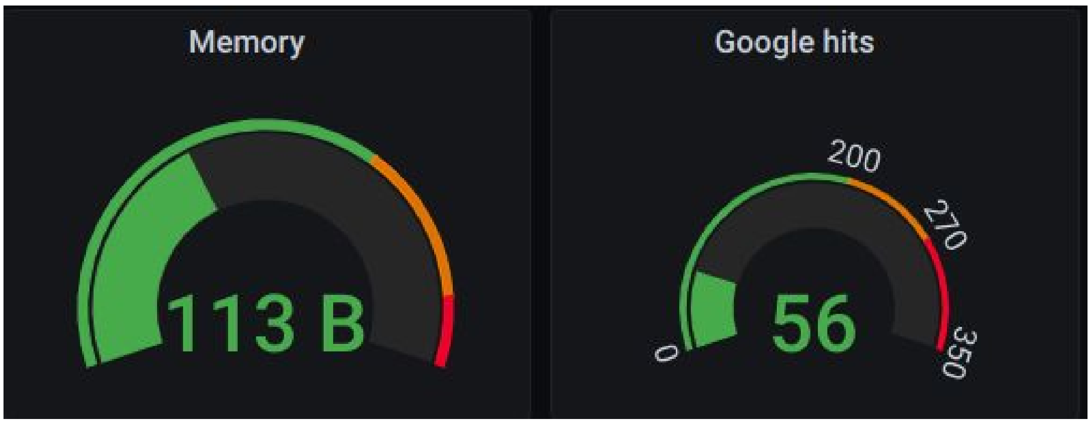




2.Guage

Gauge is a single value panel that can repeat a gauge for every series, column or row.

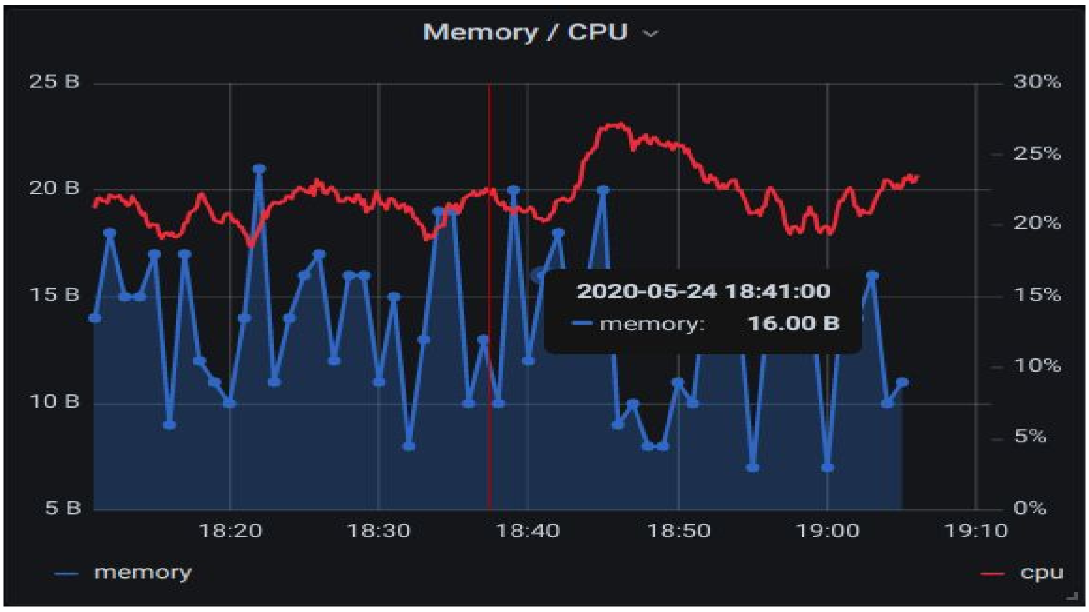
Examples :



3.Graph (Line graph OR Bar Graph)

This visualization is the most-used in the Grafana ecosystem. It can render as a line, a path of dots, or a series of bars.

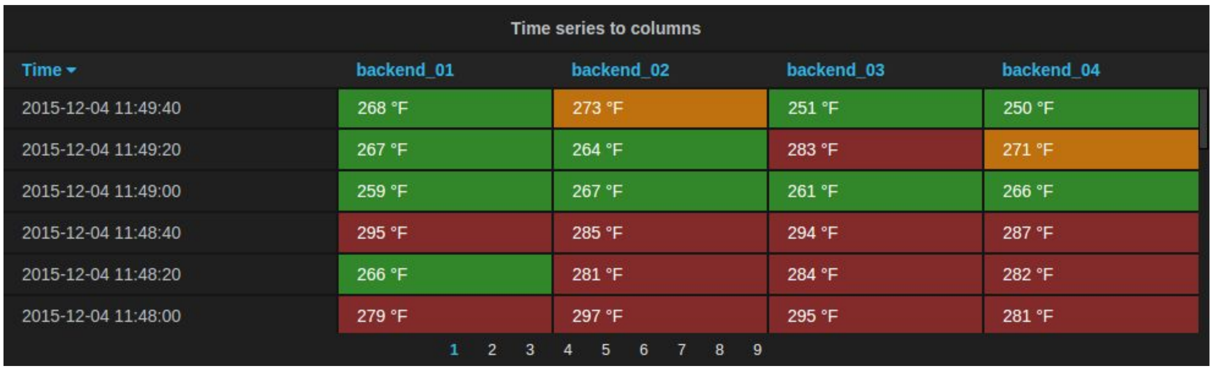
Examples :



4.Table

The table panel is very flexible, supporting multiple modes for time series and for tables, annotation, and raw JSON data. This panel also provides date formatting, value formatting, and coloring options.

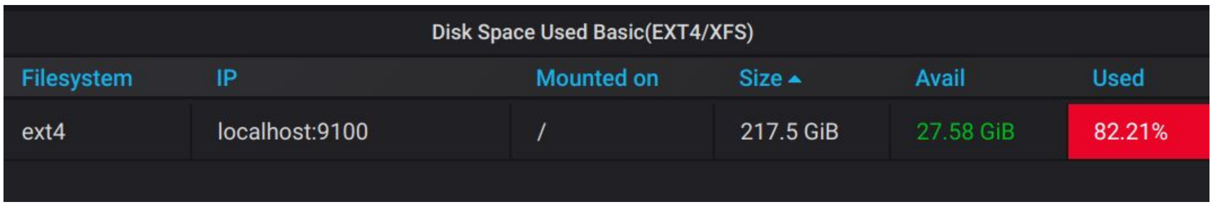
Examples :



Time series to columns

Time ▾	backend_01	backend_02	backend_03	backend_04
2015-12-04 11:49:40	268 °F	273 °F	251 °F	250 °F
2015-12-04 11:49:20	267 °F	264 °F	283 °F	271 °F
2015-12-04 11:49:00	259 °F	267 °F	261 °F	266 °F
2015-12-04 11:48:40	295 °F	285 °F	294 °F	287 °F
2015-12-04 11:48:20	266 °F	281 °F	284 °F	282 °F
2015-12-04 11:48:00	279 °F	297 °F	295 °F	281 °F

1 2 3 4 5 6 7 8 9



Disk Space Used Basic(EXT4/XFS)

Filesystem	IP	Mounted on	Size ▴	Avail	Used
ext4	localhost:9100	/	217.5 GiB	27.58 GiB	82.21%

Suggestion: Data we can include in our Fossology Dashboard

No	Data	Data Source	Type Of visualization	Remark / Details
1.	No. of unique upload hashes in FOSSoloy	***Pfile*** table	Guage	Or Simple Stats
2.	No of time agent run	***ars_master*** table	Line graph	

Some Other Notes :