Prometheus

Ravindra Kudache

Monitoring using Prometheus and Grafana

Agenda

1. Why monitoring?
2. Pillars of Observability
3. What is Prometheus?
4. What is Grafana?
5. How to setup local monitoring stack?
6. Demo Pull Metrics
7. Demo Push Metrics
8. How is Monitoring done in Oracle?
9. Questions

Pillars of Observability

There are 3 pillars of observability:

1. Metrics (Prometheus, Graphite, InfluxDB, Splunk)
2. Logs (Lumberjack, Splunk, Elastic search)
3. Tracing (Opentracing.io - Jaeger)

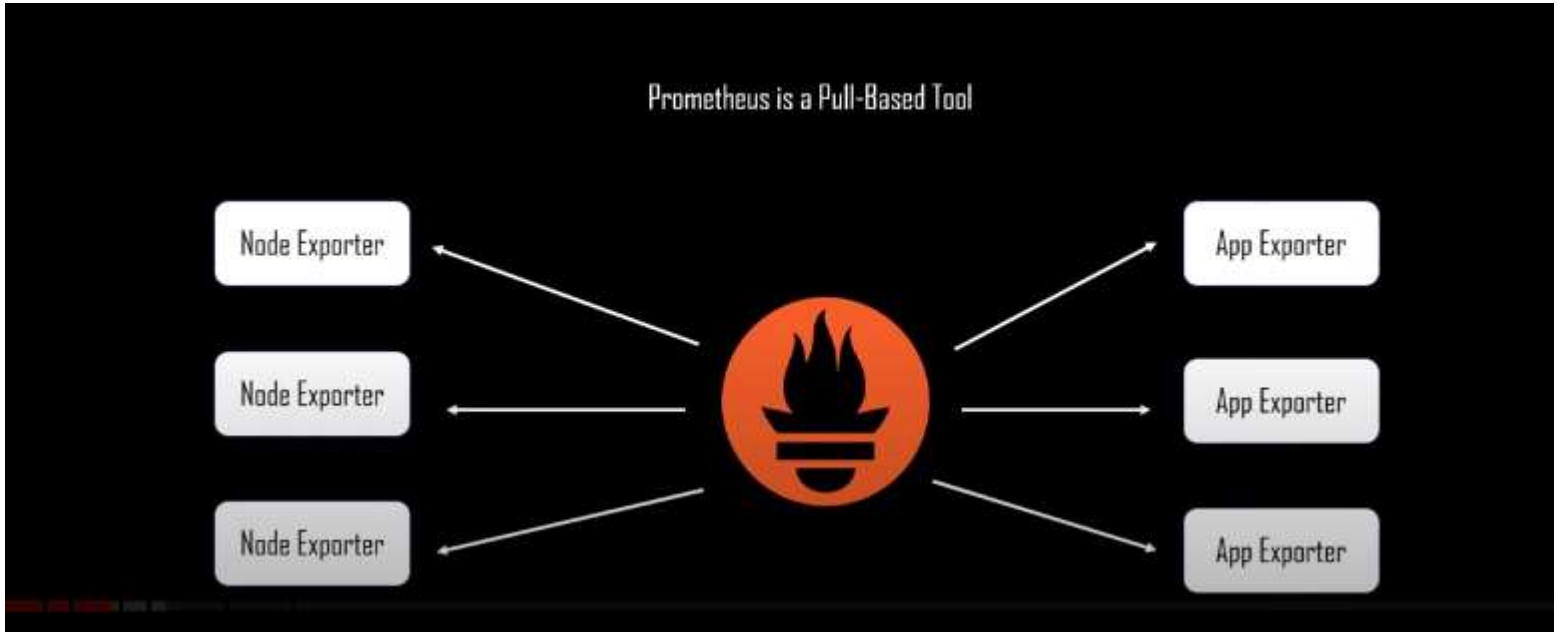
What is Prometheus?

- Prometheus is tool that you can use to monitor always running services as well as one-time jobs (batch jobs)
- Built on Go, by Soundcloud
- It persists these metrics in time-series db and you can query the metrics.
- Time series are defined by its metric names and values. Each metric can have multiple labels.
- Querying a particular combination of labels for a metric, produces a unique time-series.
- Metric is represented as-

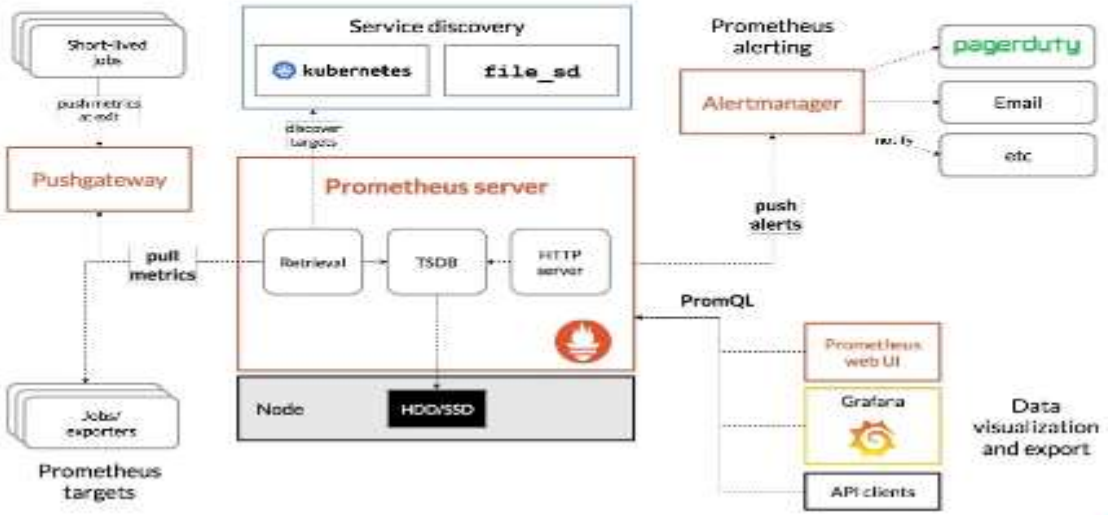
<metric name>{<label name>=<label value>, ...}

What is Prometheus?

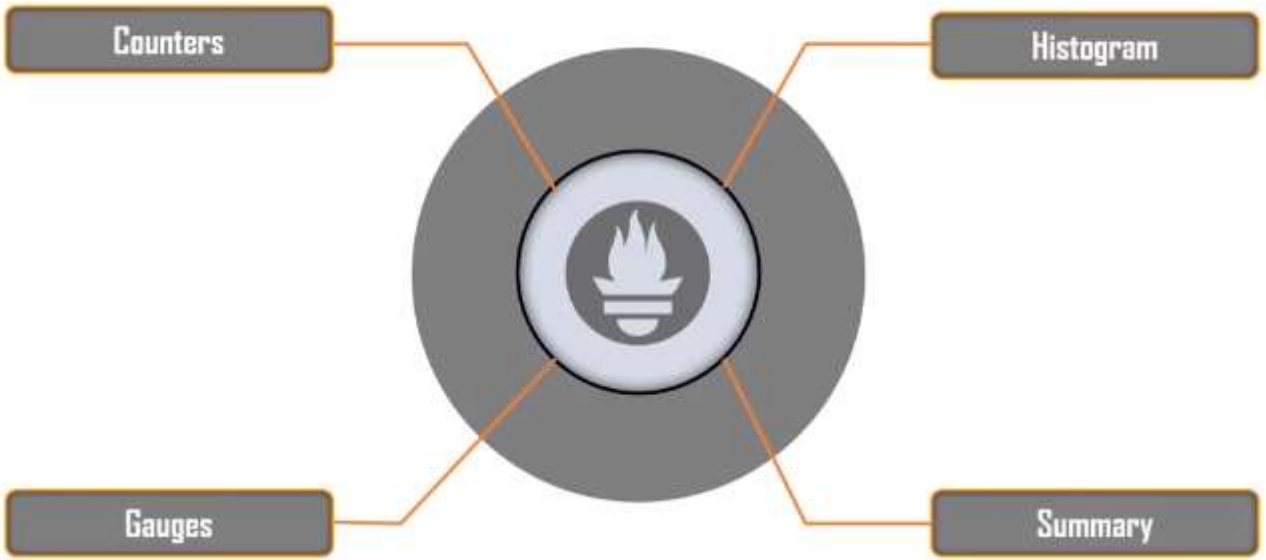
- Prometheus basically supports two modes:
- Pull - Here it is responsibility of Prometheus to pull metrics. An agent will periodically scrap metrics off configured end-point micro-services.
- Push - Here it is responsibility of the end-points to push metrics to an agent of Prometheus called Pushgateway. This is usually used for one time batch jobs

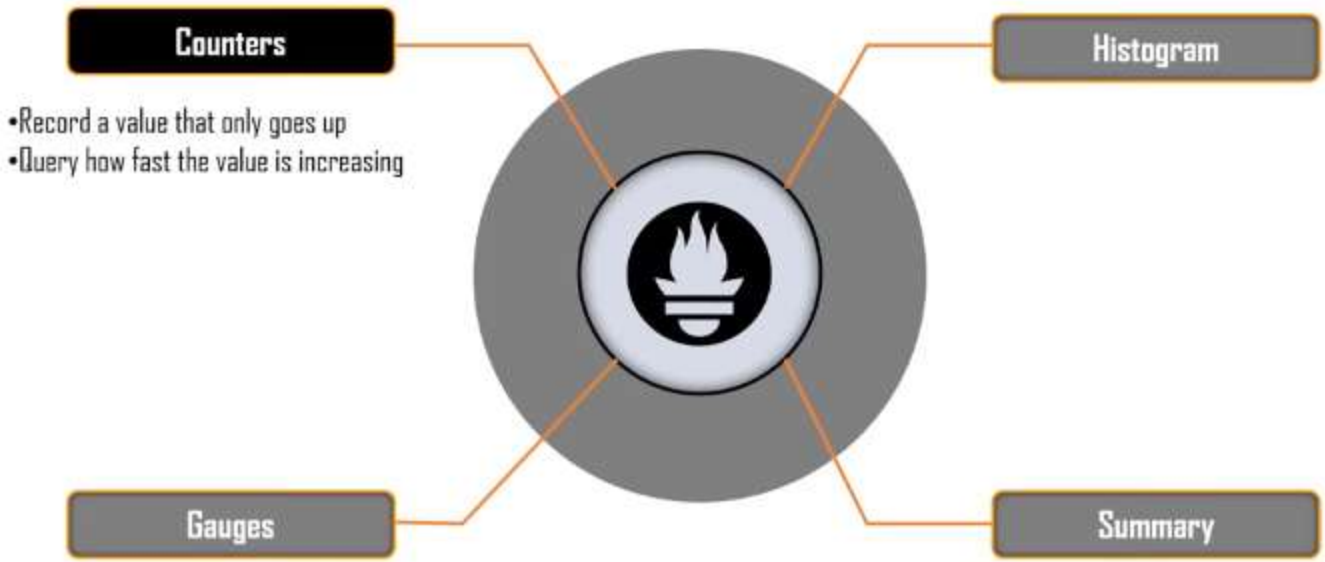

Prometheus is a Pull-Based Tool
Node Exporter
Node Exporter
Node Exporter
App Exporter
App Exporter
App Exporter

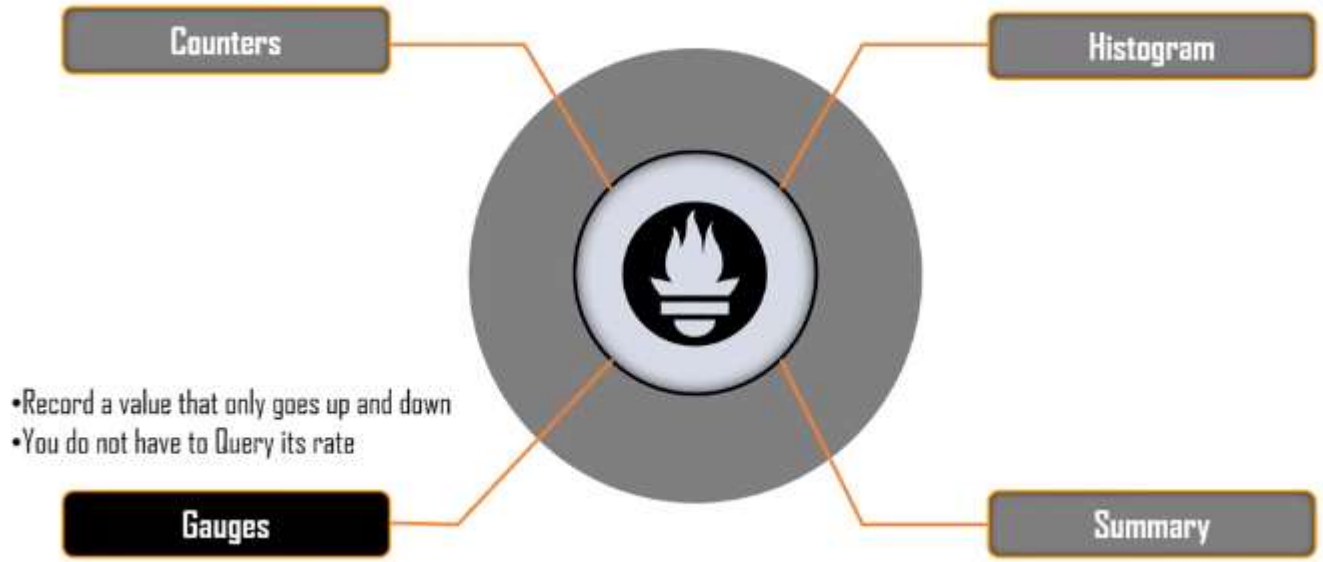
What is Prometheus?

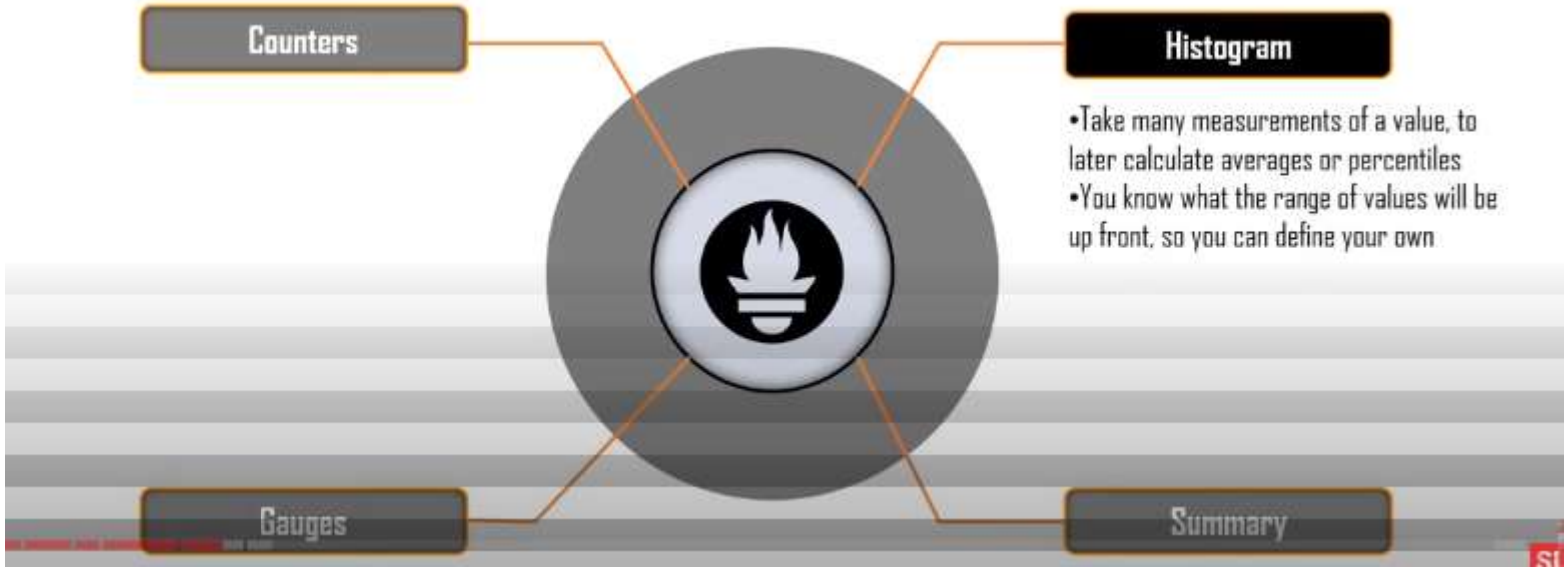


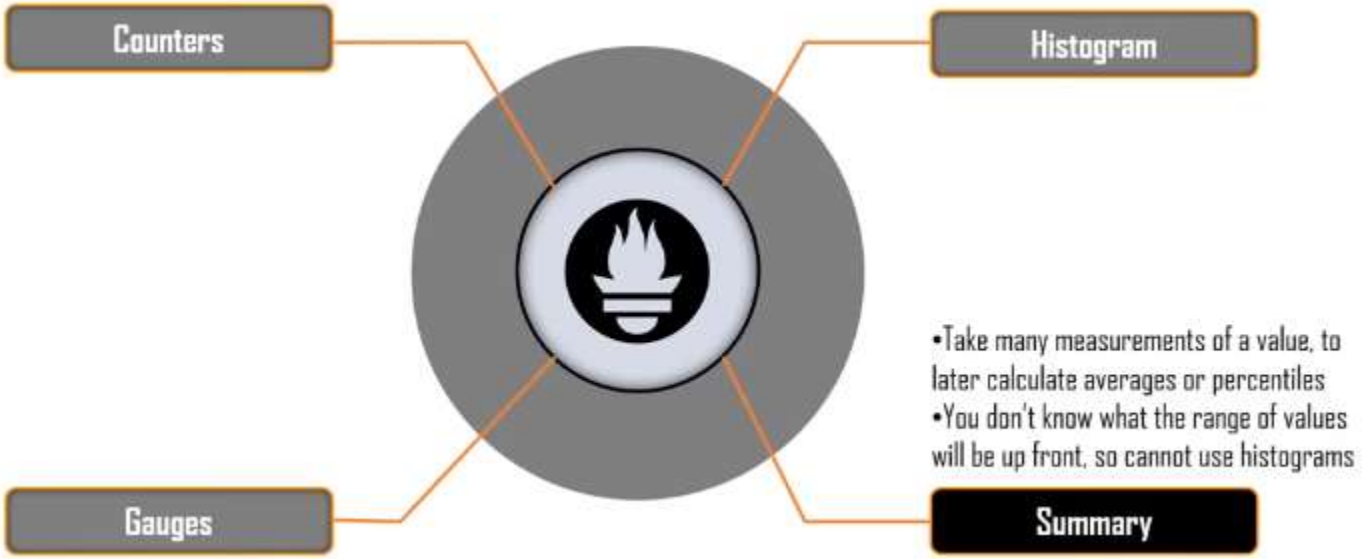
PROMETHEUS METRICS & ITS TYPES




Counters
•Record a value that only goes up
•Query how fast the value is increasing
Histogram
Gauges
Summary


Counters
Histogram
•Record a value that only goes up and down
•You do not have to Query its rate
Gauges
Summary


Counters
Histogram
•Take many measurements of a value, to later calculate averages or percentiles
•You know what the range of values will be up front, so you can define your own
Gauges
Summary



- Take many measurements of a value, to later calculate averages or percentiles
- You don't know what the range of values will be up front, so cannot use histograms

What is Prometheus?

- Metric types
- Counter - Int value that only increases or can be reset to 0.
- Gauge - single numerical value that can arbitrarily go up and down.
- Histogram - A histogram samples observations and counts them in configurable buckets. It also provides a multiple time series during scrape like sum of all observed values, count of events.
- Summary - a summary samples observations (usually things like request durations and response size) over a sliding time window.

What is Prometheus?

- Histograms and summaries both sample observations, typically request durations or response sizes. They track the number of observations and the sum of the observed values, allowing you to calculate the average of the observed values

What is Prometheus?

Instrumentation is adding sending metrics on some business logic from your microservice. For example

```
import io.prometheus.client.Counter;
class YourClass {
  static final Counter requests = Counter.build()
    .name("requests_total").help("Total requests.").register();

  void processRequest() {
    requests.inc();
    // Your code here.
  }
}
```

https://github.com/prometheus/client_java

What is Prometheus?

To expose the metrics used in your code, you would add the Prometheus servlet to your Jetty server.

Add dependency on 'io.prometheus.simpleclient' and add code below to start your metrics endpoint

```
Server server = new Server(1234);
ServletContextHandler context = new ServletContextHandler();
context.setContextPath("/");
server.setHandler(context);
context.addServlet(new ServletHolder(new MetricsServlet()), "/metrics");
```

You can also expose metrics using spring-metrics

WHY PROMETHEUS?

✓ Records any purely numeric time series

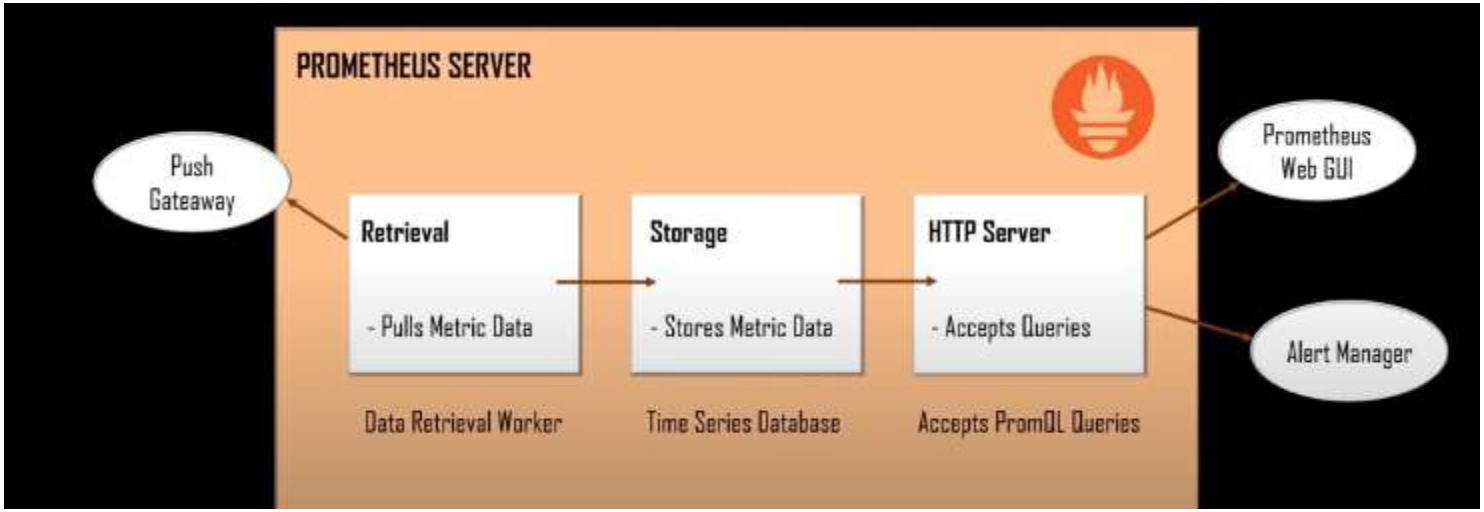
✓ Designed for Reliability

It can monitor both:

Machine-centric and
Highly dynamic service-oriented architectures


It records real-time metrics in a **time series database** built using a HTTP pull model


Applications
Exception
Memory Usage
Server CPU
Storage Spikes


PROMETHEUS SERVER
Push Gateaway
Retrieval
- Pulls Metric Data
Storage
- Stores Metric Data
HTTP Server
- Accepts Queries
Prometheus Web GUI
Alert Manager
Data Retrieval Worker
Time Series Database
Accepts PromQL Queries

What is Grafana?

Grafana is a stand-alone tool that let's you visualize your data. It can be combined with a host of different sources like – Prometheus, AWS CloudWatch, ElasticSearch, Mysql, Postgres, InfluxDB and so on. More on the supported sources .

Grafana lets you create dashboards that monitor different metrics.

You can configure alerts that can integrate with email, slack, pager duty.

What Is Grafana?

Grafana is a database analysis and monitoring tool. It allows you to create dashboard visualizations of key metrics that are important to you. Grafana has a thriving community of enthusiasts who share reusable dashboards.

Grafana supports a huge number of data sources. And, since the application is open source, you can be sure that the moment a new data source has been released, someone out there is adding support for it. The most common use case of Grafana is displaying time series data, such as memory or CPU over time, alongside the current usage data.

You can host Grafana yourself, use the managed service in AWS or get the creators to host it for you. Grafana runs as a process on your computer or server, and you access the interface through your browser. Your dashboard can display your data as single numbers, graphs, charts, or even a heat map. Below is an example dashboard set up through the Grafana web interface:


Memory / CPU
logins
Memory
80 B
Sign ups
275
Logouts
161
Sign outs
279
server requests
client side full page load
70 GB
60 GB
60 GB
30 GB
40 GB
83 GB
44 GB
47 GB
94 GB
4 GB
73 GB
44 GB
19 GB
22 GB
20 GB
81 GB
sda1
sda2
sda3
sda4
sda5
sda6
sda7
sda8
sda9
sda10
sda11
sda12
sda13
sda14
sda15
sda16

Installing Grafana Locally

The instructions for installation depend on where you plan to run Grafana. I normally run applications like this locally to get a feel for how it works before deploying remotely. Grafana provides comprehensive instructions for wherever you want to deploy it.

If the installation was a success, then Grafana should be available through your web browser by default on http://localhost:3000, and your screen should look like the image below. The default username is **admin** and the default password is also **admin**. You will be asked to change your password after logging in.

END