

## The Future of Observability: Where Prometheus Fits in the Landscape !!



# Prometheus

**Prometheus is an open-source monitoring system and time-series database. It was developed by SoundCloud in 2012 and is now maintained by the Cloud Native Computing Foundation (CNCF).**

### **Why do we need monitoring tools even?**

*In today's fast-paced and distributed environments, applications consist of various interdependent services, infrastructure components, and networks that work together to provide functionality. This complexity makes it difficult to pinpoint issues, optimize performance, and maintain uptime without a thorough monitoring solution. Monitoring tools offer visibility into these systems by gathering, analyzing, and visualizing essential metrics, logs, and events, which enables teams to make informed decisions and tackle issues proactively.*

*A primary reason monitoring tools are essential is the increasing demand for applications and services that are always available. Downtime can lead to lost revenue, reduced user trust, and harm to a*

*company's reputation.*

*Monitoring tools help reduce these risks by delivering real-time insights into system performance and notifying teams of potential problems before they develop into outages.*

*For example, monitoring tools can identify spikes in CPU usage, memory leaks, or slower response times, allowing teams to investigate and resolve issues quickly.*

### **Let's take an example:**



*In an e-commerce platform hosting a flash sale, a sudden surge in user traffic can overwhelm servers, slow down database queries, or even cause complete outages if not addressed quickly.*

*Monitoring tools like Prometheus or Datadog continuously track essential metrics such as server CPU usage, memory consumption, and response times. These tools gather real-time data to identify bottlenecks and send alerts before problems escalate.*

*For example, if database latency rises or web server responses lag, the operations team can be alerted right away to scale resources or resolve issues.*

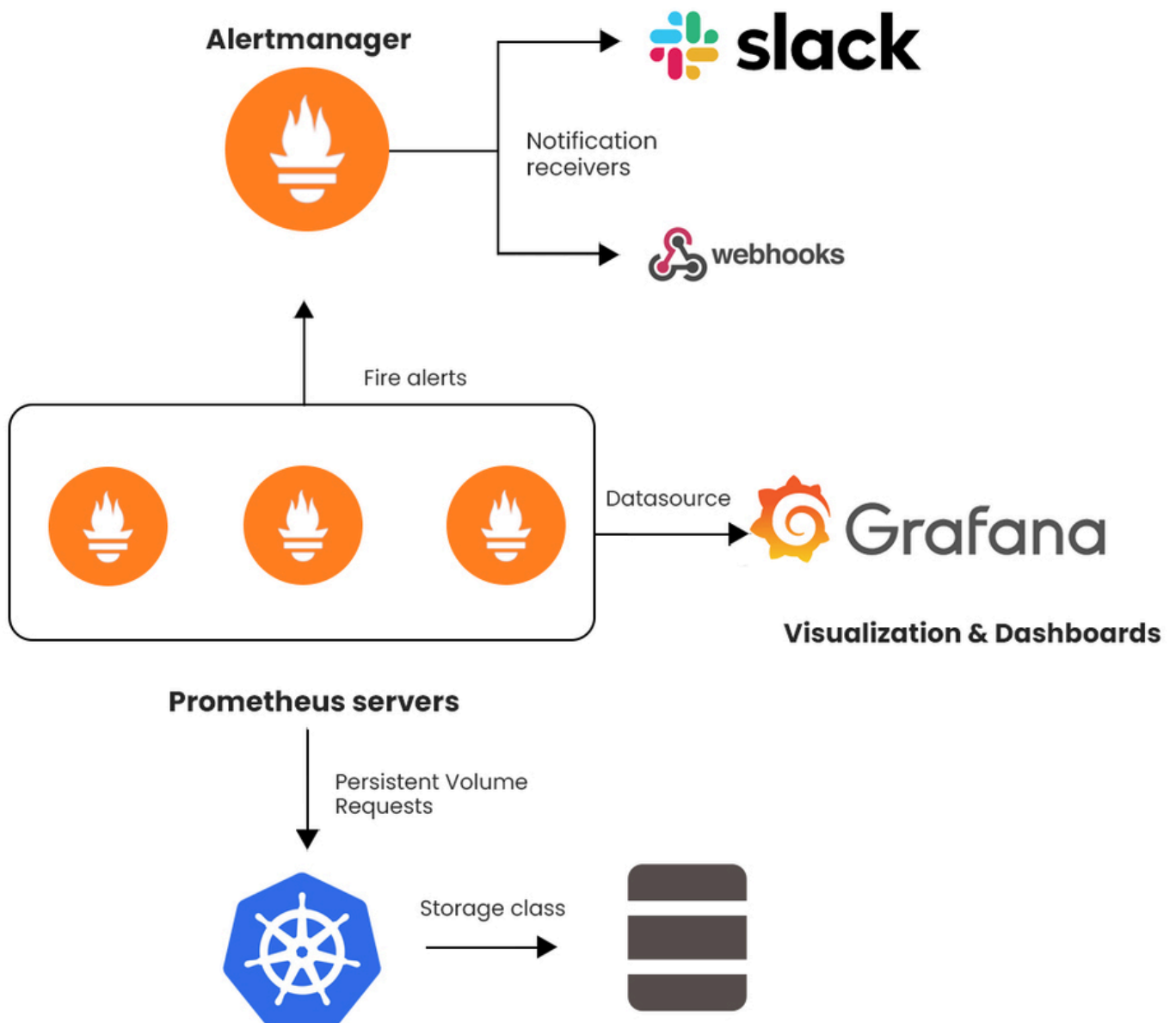
*Similarly, in a streaming service like Netflix, users may face buffering due to overloaded regional servers. Monitoring tools assess content delivery network (CDN) latency and bandwidth in real-time, pinpointing trouble spots and rerouting traffic to ensure smooth streaming quality. These instances illustrate how monitoring tools not only assist in troubleshooting but also actively prevent disruptions, providing users with an optimal experience while keeping the system stable.*

So, To troubleshoot the issue you check backward to see where the cause is. So you ask yourself these typical questions:

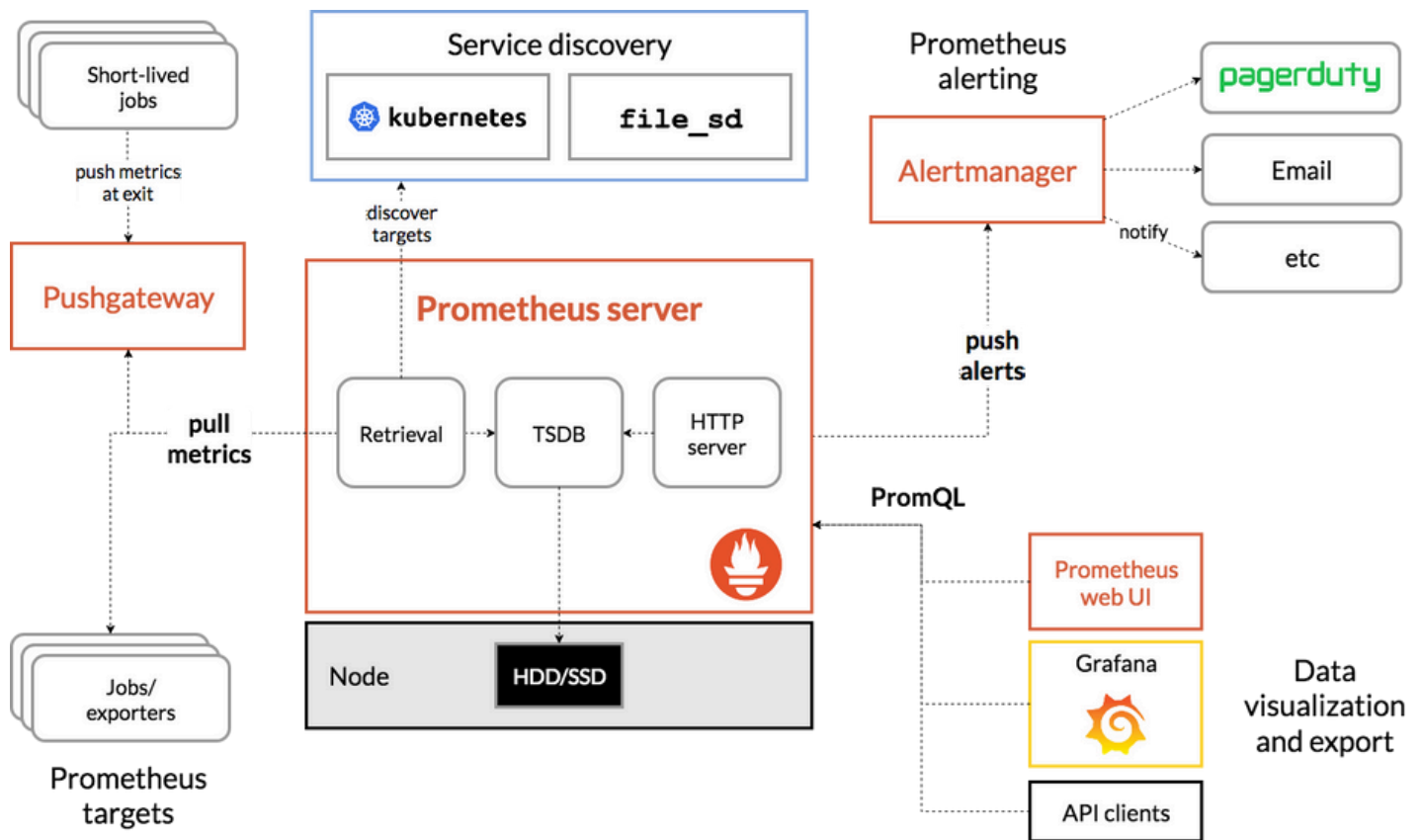
- Is frontend running?
- Is the service running?
- Is Backend running?
- Did any exceptions occur?
- Is the database running?

So to minimise these extra efforts we required a tool that constantly monitor all the services and sends an alert when a particular service crash. Also, identify the cause of failure before it occurred. For example, It should send an email to the Administrator when the CPU reaches 70% of its limit so that administrator can get a warning before the failure occurs in the application

## How Does it Work?



The default port number for Prometheus is 9090, and its architecture revolves around three main components:



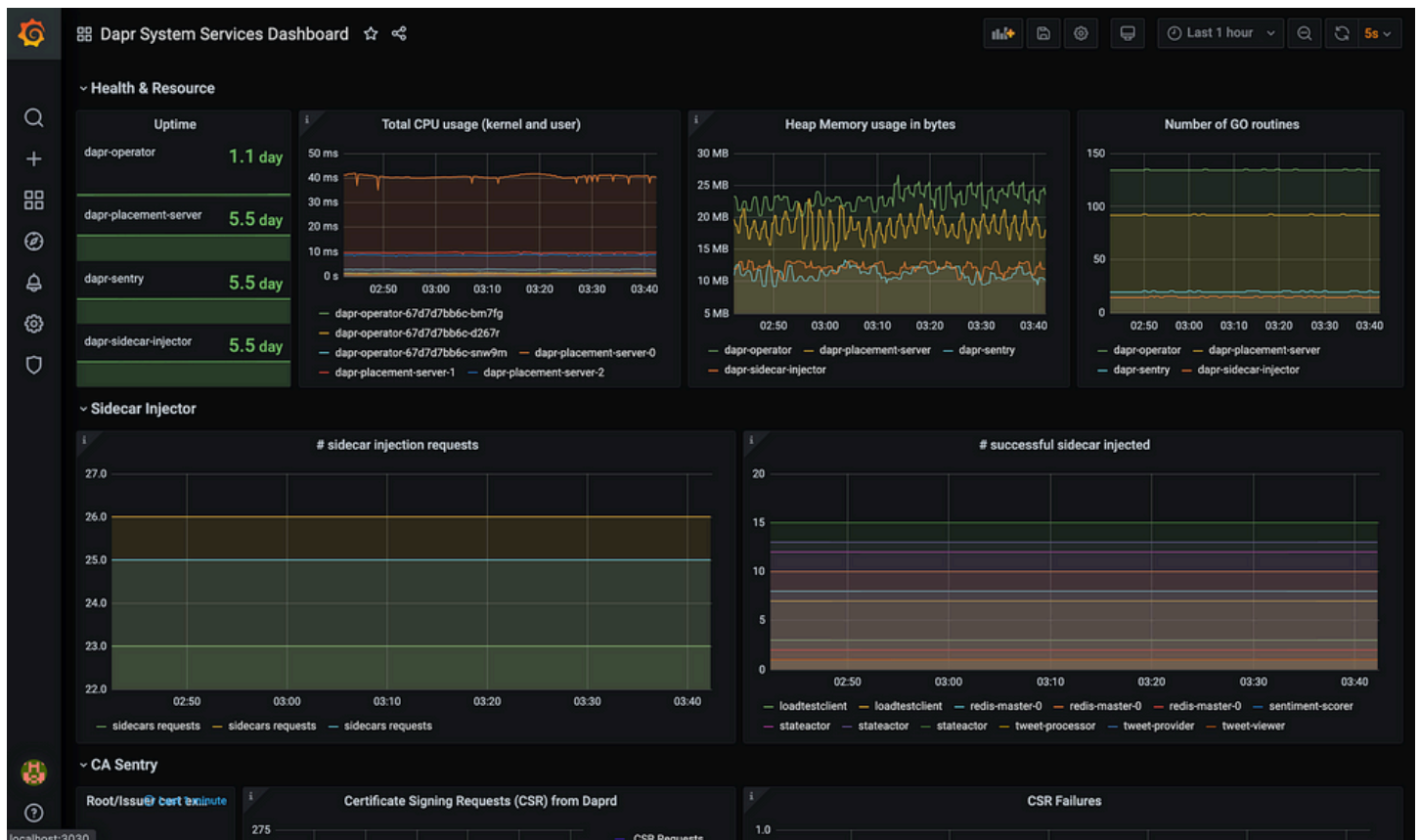
1. **Retrieval:** This part is tasked with collecting metrics from specified targets using a pull-based approach.
2. **Time-Series Database (TSDB):** This is where the gathered metrics are stored and managed for efficient querying and analysis.
3. **HTTP Server:** This component offers a web interface and API, allowing users to interact with Prometheus and utilize its powerful query language, PromQL, for querying and visualization.

## What does Prometheus monitor?

Here are some examples of the targets that Prometheus monitors.

- **Linux server**
- **Apache server**
- **Single standalone application**
- **Services such as DB**

## What are metrics?



Metrics are human-readable data that have two main attributes: TYPE and HELP.

The HELP attribute explains the reason or context of the error, while the TYPE attribute categorizes the metric.

There are three main types of metrics:

1. **Counter:** This tracks the total number of times an event has occurred, such as how many times a specific process has run or the total CPU usage.
2. **Gauge:** This monitors values that can vary, such as current memory usage or CPU load.
3. **Histogram:** This measures distributions over time, focusing on aspects like duration or size. For example, it can log the size of incoming requests or the time taken for specific operations.

## How it collects the data from Targets?

Prometheus pulls metrics data from the targets from HTTP endpoints.

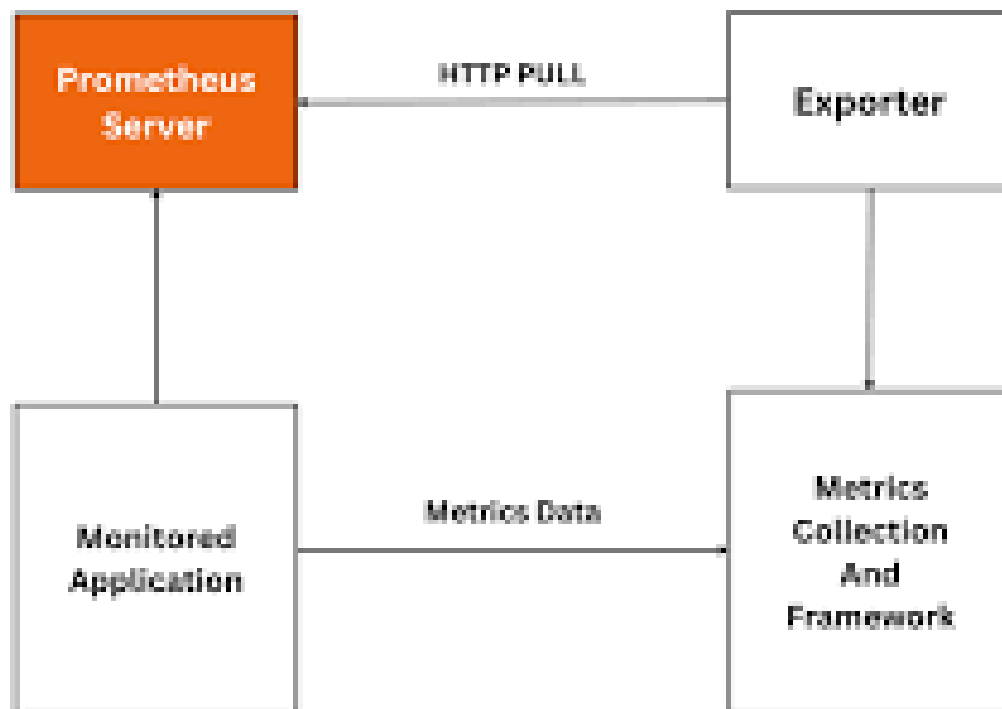
By default, it scraps data from hostaddress/metrics endpoint.

Data at the target must be in the format that Prometheus understands.

## What is an Exporter?

Some servers by default expose the data to the /metrics endpoint so we don't need to do extra work. But for the services that do not have native Prometheus endpoints, we required an extra component to do the work so that is *Exporter*.

The exporter is a script that pulls the data from your target and converts it to the format understood by Prometheus and exposes this data on its own /metrics endpoint where Prometheus can pull them.



## Problem with Push Mechanism?

As we aware prometheus uses pull mechanism to get the data from the HTTPS endpoints.

But is there any problem with push mechanism , why we choose pull over that push mechanism. So basically:

## Pull has advantage over Push because:

1. **Network Congestion:** The continuous pushing of metrics by various microservices creates a lot of network traffic, which can overload the network and negatively affect performance.
2. **Infrastructure Overhead:** Keeping up with and scaling the infrastructure to manage the constant influx of pushed data can be expensive and resource-heavy.
3. **Installation and Configuration Complexity:** Each service needs a dedicated agent or daemon installed and configured to push metrics, which adds to the operational burden.

### **How Prometheus Solves This Problem:**

**Pull Mechanism:** Rather than having services push data, Prometheus pulls metrics from endpoints that each service exposes.

### **This method:**




1. **Reduces Network Load:** By centralizing data collection, Prometheus lessens network traffic and prevents network overload.
2. **Simplifies Infrastructure:** It removes the necessity for agents on every service, thereby decreasing infrastructure complexity and management demands.
3. **Provides Service Health Checks:** The pull mechanism automatically checks the availability and health of each service. If a service is down or not responding, Prometheus won't be able to retrieve metrics, signaling a potential problem.

### **Addressing Short-Lived Jobs:**

**Pushgateway:** For short-lived jobs, such as cron jobs or scheduled tasks, a Pushgateway serves as an intermediary. These jobs send their metrics to the Pushgateway, which then makes them available to Prometheus for collection. This setup enables Prometheus to gather metrics from short-lived processes that may not be present during the usual pull interval.

---



Thank you for reading! If you found this post helpful, feel free to connect with me on LinkedIn  or follow me on Twitter  for more insights on Kubernetes and cloud-native technologies. .