

What is Prometheus?

- Prometheus is an **open source monitoring solution** written in **Go** that ***collects metrics data and stores that data in a time series database***.
- It was originally built by **SoundCloud** in 2012 and became part of the **Cloud Native Computing Foundation (CNCF)** in 2016.
- It uses PromQL, a powerful query language for querying your time series data.

How does Prometheus work?

- Prometheus scrapes metrics data from HTTP endpoints and then pushes that data into a database that uses a **multidimensional model.**
- **Data collection and retrieval:** Prometheus follows **a pull-based model**, where it periodically scrapes data from the targets (applications, services or infrastructure components) that are instrumented with its client libraries. The targets expose metrics and endpoints, allowing Prometheus to gather relevant information.
- **Data storage**: The collected metrics are stored in a time-series database, providing a historical record of system performance over time.
- **Service discovery:** Prometheus utilizes discovery mechanisms to ensure that **new instances are automatically detected and monitored** without manual intervention.

What is PromQL?

- PromQL, which is short for Prometheus Query Language, is a functional tool which features allows you to select and aggregate time series data.
- It's a flexible, powerful language that allows you to slice and dice your data however you need.

Prometheus features

- **Multi-dimensional data**: Prometheus uses a multi-dimensional data model to represent time-series data. This model allows flexibility in organizing and querying metrics based on various dimensions

like job, instance, and labels, enabling detailed analysis.

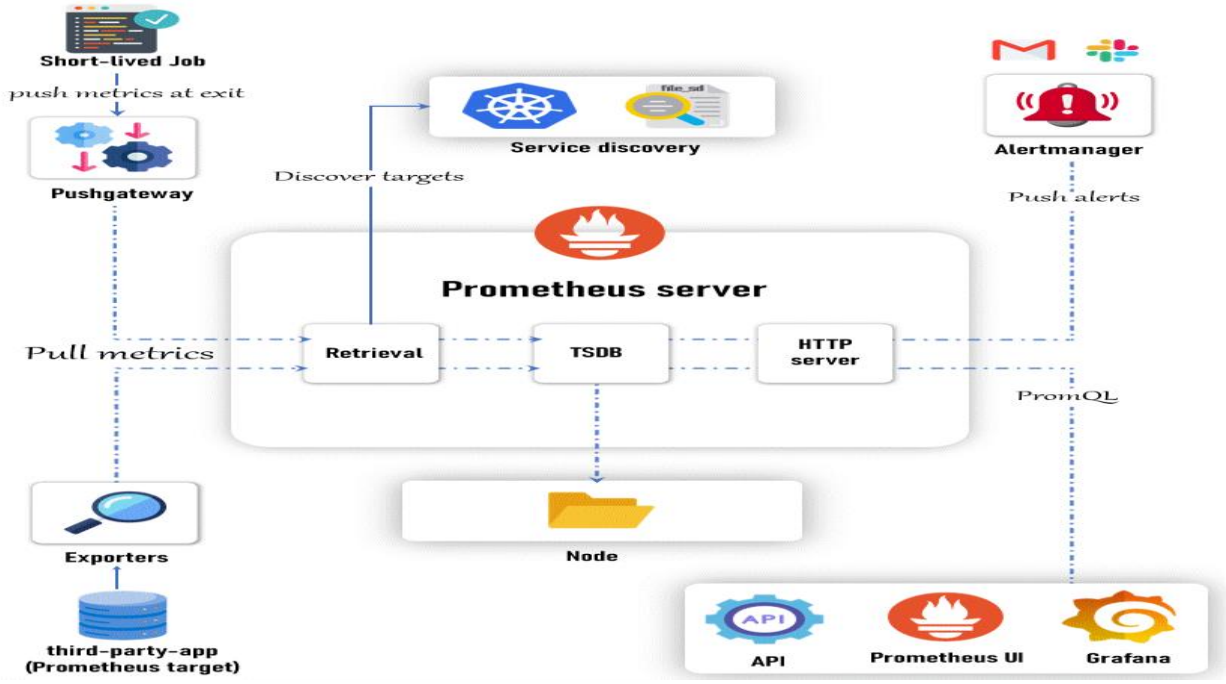
- **PromQL**: As discussed, Prometheus uses a powerful query language to slice and dice collected time series data, allowing flexibility in monitoring diverse systems and services.

- **Alerting rules**: Prometheus allows you to **define alerting rules** based on specified conditions. If the system detects that a **predefined condition is met**, it will trigger an alert, letting teams know about potential issues before they impact users.

- **Data visualization through integrations**: Prometheus is often used with other tools, like **Grafana** (an open-source analytics and monitoring platform). Grafana allows users to create visually appealing dashboards and reports based on the data collected by Prometheus. Additionally, Prometheus integrates seamlessly with **Kubernetes**, making it a preferred choice for monitoring containerized environments.

- **Scalability and federation**: Prometheus can be deployed in a federated setup, enabling multiple Prometheus instances to collaborate. This allows for scalability making the platform suitable for large and distributed architectures.

Architecture Diagram



Core Components

- **Prometheus Server** → Scrapes metrics from targets, stores them in a local time-series database

(TSDB), and executes queries with PromQL.

- **Targets & Exporters** → Applications expose metrics via HTTP endpoints; exporters provide metrics for systems like MySQL, Node, HAProxy.
- **Service Discovery** → Automatically finds targets using Kubernetes, Consul, or static configs.
- **Alertmanager** → Handles alerts, supports deduplication, grouping, and routing to email, Slack, PagerDuty.
- **Push Gateway** → Allows short-lived jobs to push metrics instead of being scraped.
- **PromQL** → Flexible query language for aggregations, filtering, and visualization.

What are metrics?

- Metrics are **numerical measurements** in layperson terms.
- The term **time series** refers to the recording of changes over time. What users want to measure differs from application to application. For a web server, it could be request times; for a database, it could be the number of active connections or active queries, and so on.
- The core metric types:
- **Counter**: A cumulative metric that only goes up, such as the total number of HTTP requests.
- **Gauge**: A metric that can go up and down, such as current memory usage or CPU temperature.
- **Histogram**: Samples observations and counts them into configurable buckets, such as request durations.
- **Summary**: Similar to a histogram, but calculates configurable quantiles over a sliding time window.

What is Prometheus monitoring?

- An open-source monitoring and alerting toolkit that can be used for monitoring highly dynamic service-oriented architectures and hardware.

What can you monitor with Prometheus?

- **Frontend**:
- Use Prometheus to monitor application metrics like throughput (TPS) and response times. In particular, the Prometheus blackbox exporter enables you to perform an uptime check and monitor website status.

- **Backend:**
- Monitor databases, APIs, and HTTP request status on your endpoints using Prometheus. For instance, you can use the toolkit to track REST API metrics like request latency, log events, and error rate per API. You can also use Prometheus to monitor JVM applications via the JMX Exporter.
- **Servers**:
- Leverage Prometheus Blackbox Exporter to track server KPIs like average response time. In addition, you can monitor your operating system to understand your server CPU utilization or its hard disk usage. You can also use Apache Prometheus exporter to monitor Apache web server.
- **Hardware**:
- With **Prometheus Node Exporter**, you can monitor hardware and kernel metrics on Linux and any other Unix system. Some of the hardware metrics you can track include CPU usage, disk utilization, network bandwidth, and memory.
- **Infrastructure**:
- Prometheus can monitor your infrastructure and applications at multiple levels, including hosts, the application itself, and any containers. For instance, you can monitor the performance of MySQL and identify any issues with the database using the Prometheus MySQL exporter. Another example is the ability to track the throughput and response times of the Kafka load generator, Cassandra client, and Kafka consumer.

When Prometheus Fits

- **Numeric time series** → Ideal for recording metrics like CPU usage, request counts, latency.
- **Machine-centric monitoring** → Works well for servers, containers, and infrastructure.
- **Microservices monitoring** → Strong support for multi-dimensional data collection and querying.
- **Reliability under outages** → Standalone servers ensure monitoring continues even if other systems fail.
- **Quick diagnosis** → Designed to be the go-to tool during incidents

When Prometheus Does Not Fit

- **100% accuracy needs** → Not suitable for billing or per-request financial tracking.
- **Detailed transaction data** → Collected metrics may not be complete enough for compliance or accounting.

- **Long-term storage** → Requires external integrations for historical data retention.
- **Complex distributed storage** → Prometheus avoids reliance on external distributed systems, which may be a limitation for some use cases.

Alerting and Visualization

- Prometheus focuses on metric collection and evaluation, and it delegates alerting and visualization to specialized companion tools:
    - **Alertmanager**: Handles the deduplication, grouping, and routing of alerts to notification services such as Slack, PagerDuty, or email.
    - **Grafana**: The standard companion tool for Prometheus. Grafana connects to your Prometheus data source to build rich, visual dashboards and graphs.