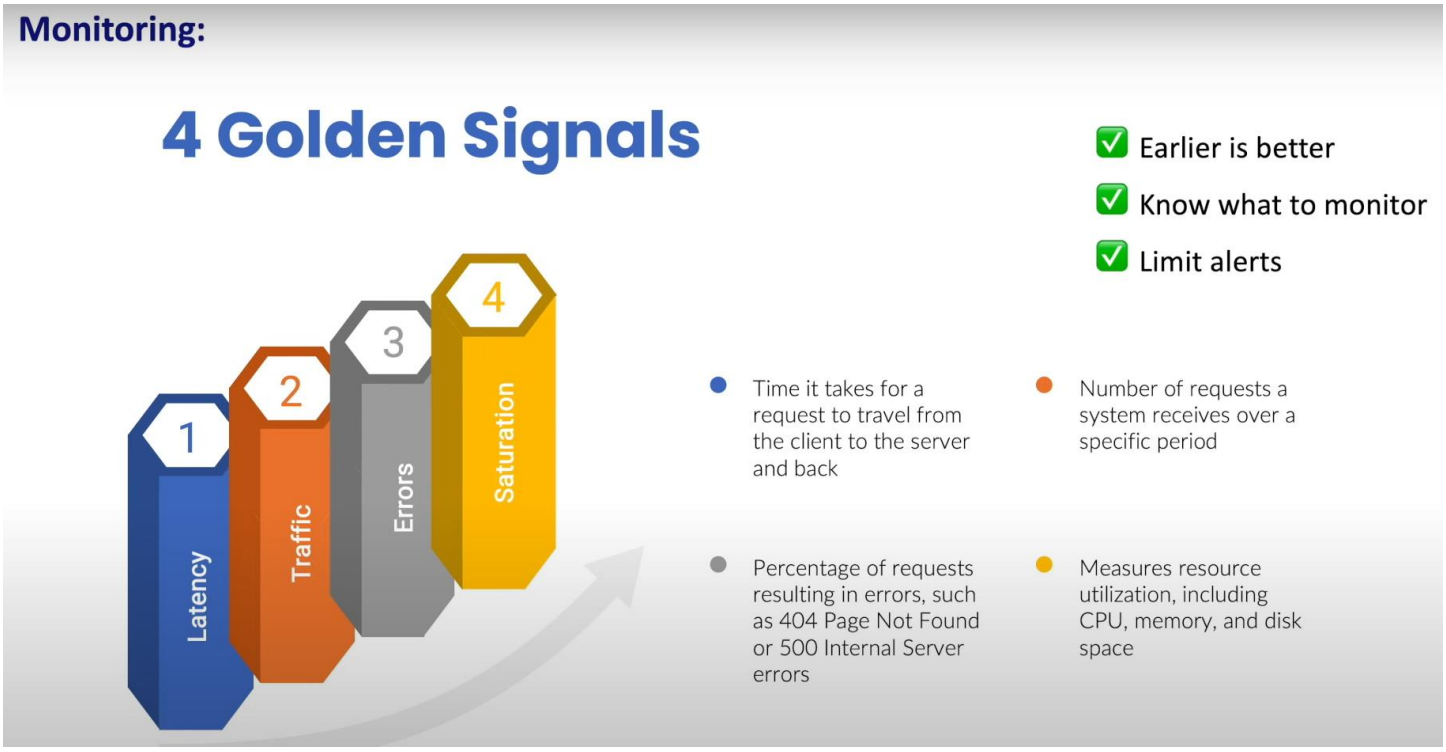
Monitoring and Observability

Introduction to Monitoring and Observability:

- **Observability** and **Monitoring** are often confused but are subtly different concepts, both crucial for identifying and fixing issues in software applications.
- **Goal**: To detect bugs early, ensure high-quality service, and improve user experience.

1. What is Monitoring?

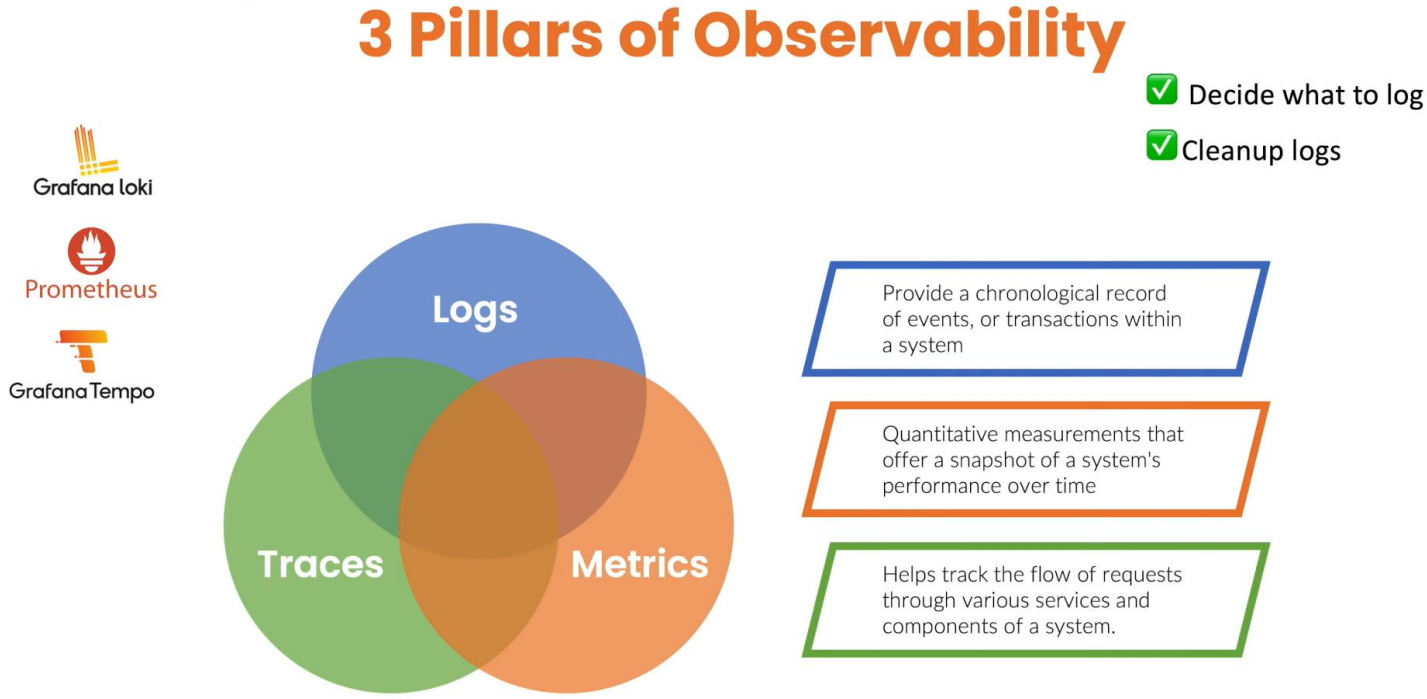
- **Definition**: Monitoring involves tracking the health and performance of an application, detecting issues, and providing alerts before problems escalate.
- **Example**:
  - In an e-commerce web application, users might face latency when searching for products due to a poorly performing query in the search service.
  - Monitoring tools (e.g., Prometheus, Grafana) track metrics like **response time** and alert when a service is running slow.



2. What is Observability?

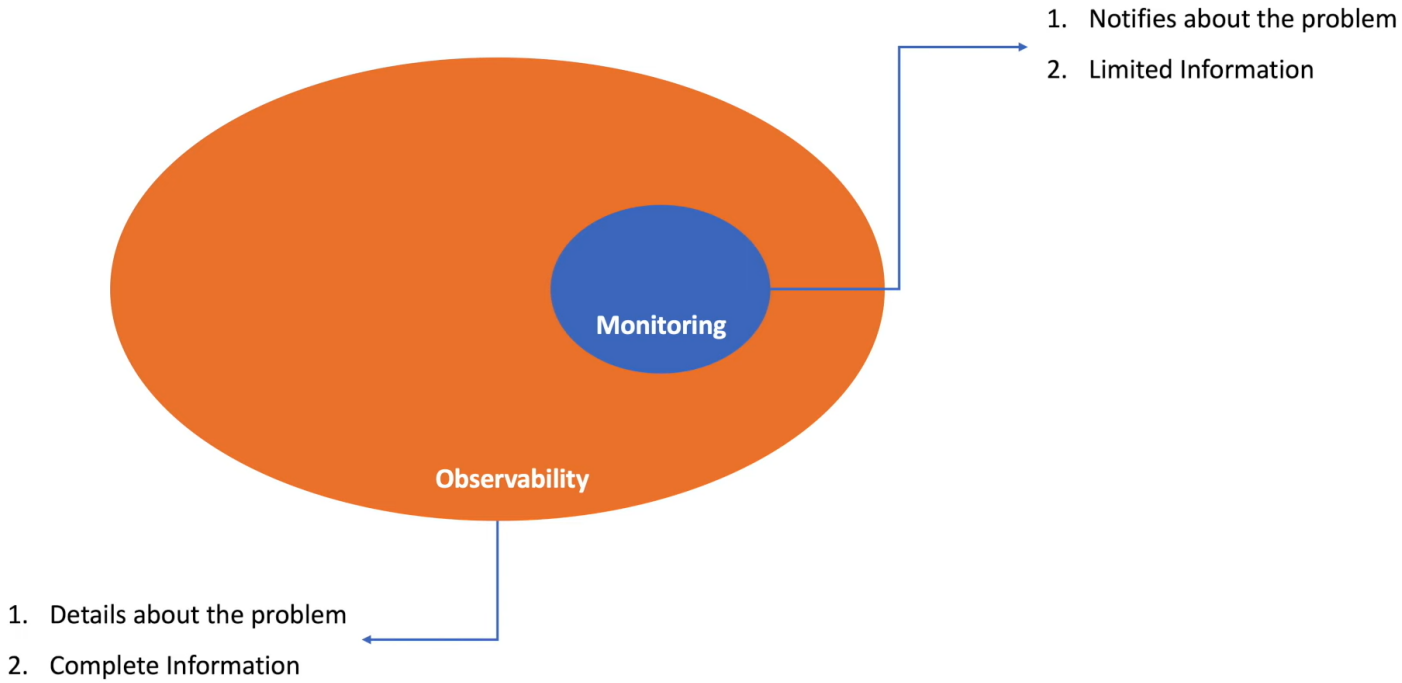
- **Definition**: Observability allows for deep investigation into the root cause of issues. It provides the data (logs, metrics, and traces) necessary to troubleshoot and resolve problems.

Observability:



- **Tools for Observability**: Elasticsearch, Prometheus, Zipkin, Tempo.

Differences:



Analogy:

- **Monitoring** is like detecting an increased heart rate in a patient via vital signs.
- **Observability** is like examining the patient's full history, activities, and symptoms to find the exact cause (e.g., an allergic reaction).

Conclusion

- **Monitoring**: Detects the problem (e.g., high response time).
- **Observability**: Provides the context to identify the root cause (e.g., which microservice is the bottleneck).
- **Together**: They help teams ensure system health, user experience, and scalability.

Summary

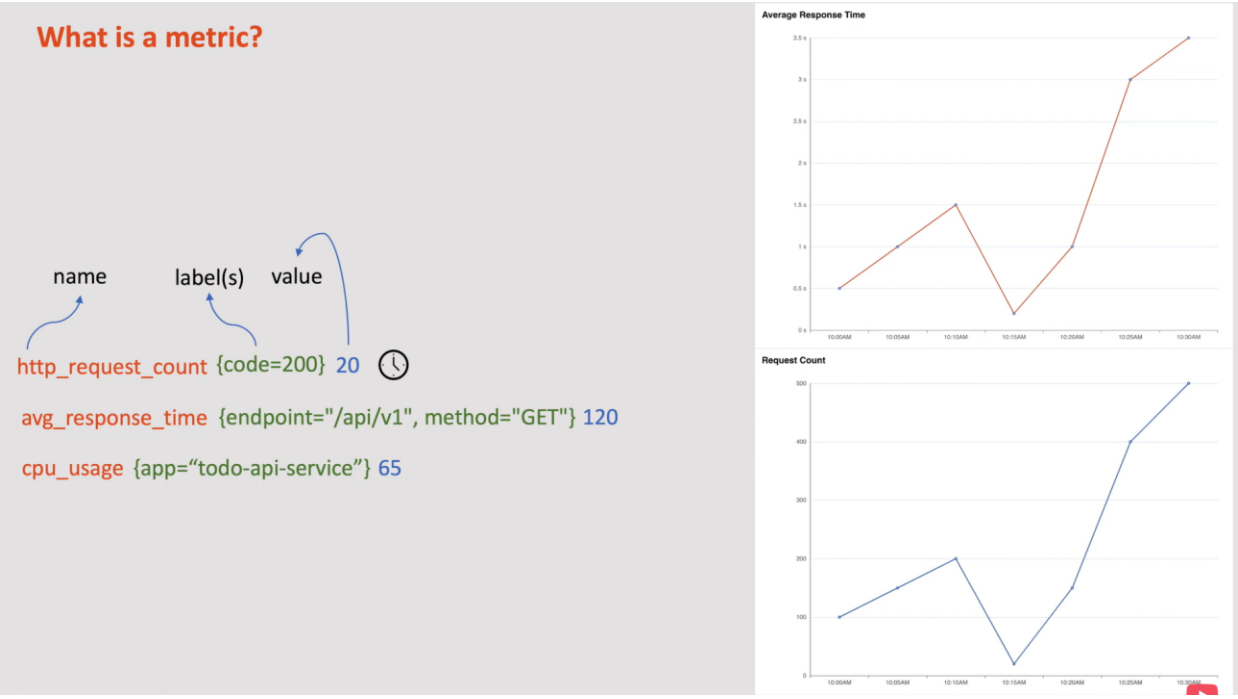
- **Monitoring** alerts you when something goes wrong.
- **Observability** helps you figure out **why** it went wrong by providing a complete set of data (logs, metrics, and traces).

<u>Prometheus Overview:</u>

- Prometheus is widely used for storing time series data and providing insights into system performance.

1. What is a Metric?

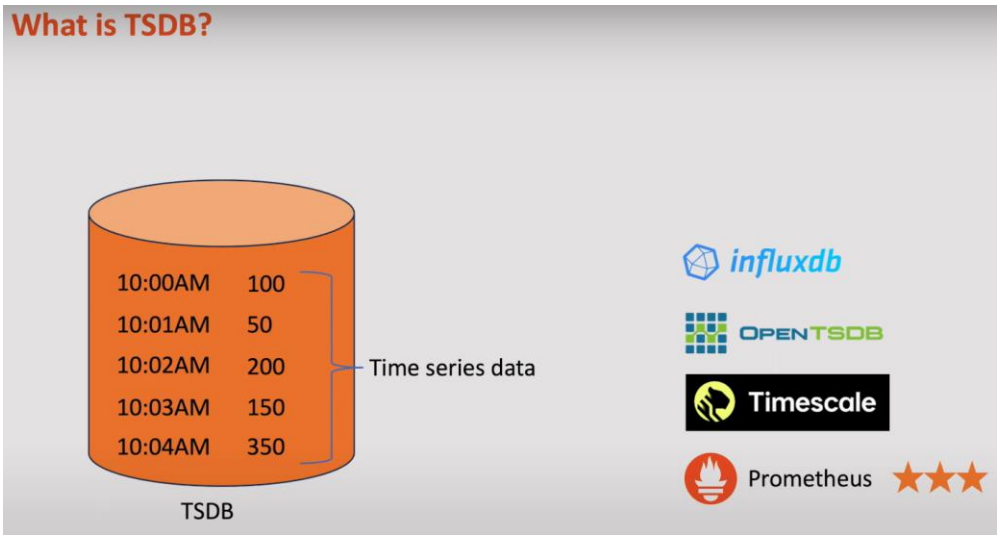
- **Definition**: A metric is a numerical measurement that tracks some aspect of a system's performance.
- **Example**:
  - A **request count** metric in a to-do application could indicate the number of requests made in a given time frame.
  - A metric consists of:
    - **Name**: e.g., http_request_count
    - **Value**: e.g., 20 requests
    - **Optional Labels**: Additional attributes to describe the metric (e.g., status_code=200).



2. What is a Time Series Database (TSDB)?

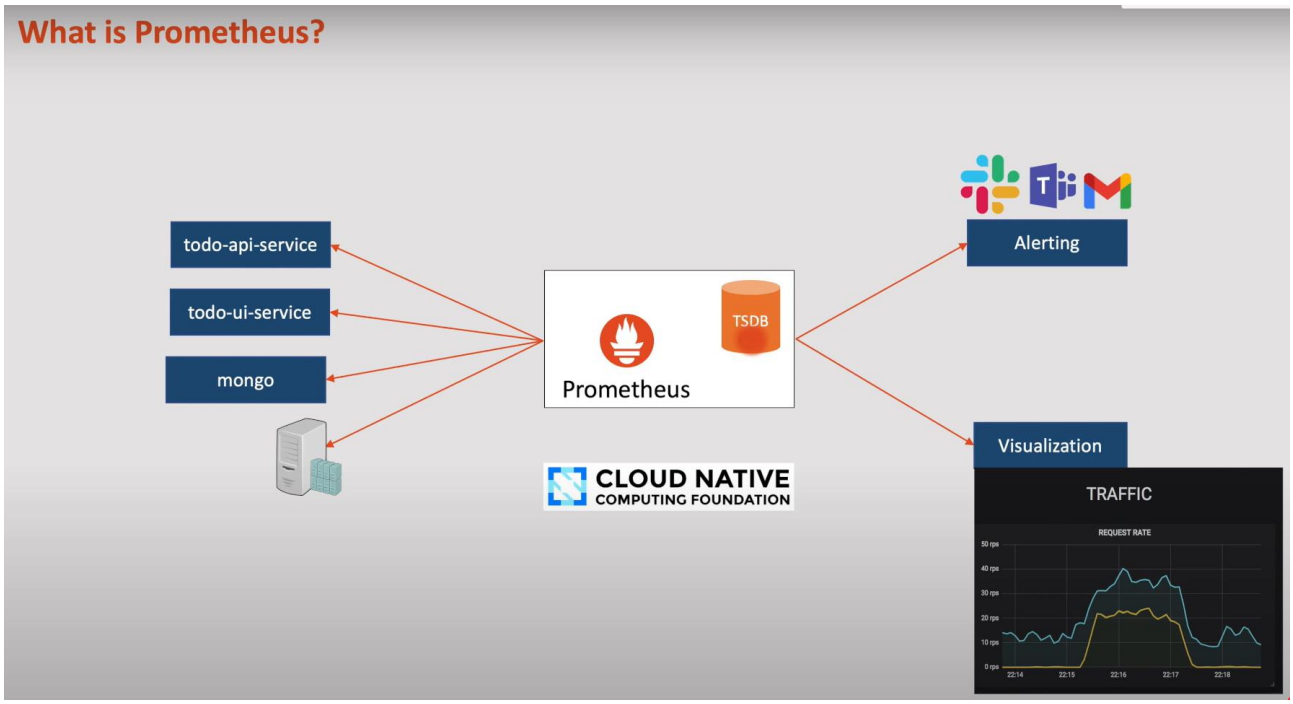
- **Definition**: A TSDB is a specialized database optimized for storing and querying time-stamped data.
- **Why TSDB?**: Traditional relational databases are not efficient for storing time series data.

- **Features**: TSDBs store data points with timestamps and associated values (e.g., request count at specific time intervals).



3. What is Prometheus?

- **Definition**: Prometheus is an open-source **observability** tool designed for collecting and storing time series data.
- **Use Cases**:
  - **Metrics Collection**: Prometheus gathers metrics from various services (e.g., applications, nodes).
  - **Centralized Metrics Storage**: Prometheus stores all metrics in a central location for easy analysis.
  - **Insights and Dashboards**: Use the data to create dashboards for monitoring system performance and behavior.
  - **Alerts**: Set up alerts for critical events like service outages or high resource usage.
- **Origin**: Developed by former Google engineers at **SoundCloud** in 2012 as an internal tool.
- **Cloud Native**: Now maintained by **CNCF** and integrated with the **cloud-native ecosystem**.
- **Native Kubernetes Support**: Prometheus provides native support for monitoring Kubernetes environments.



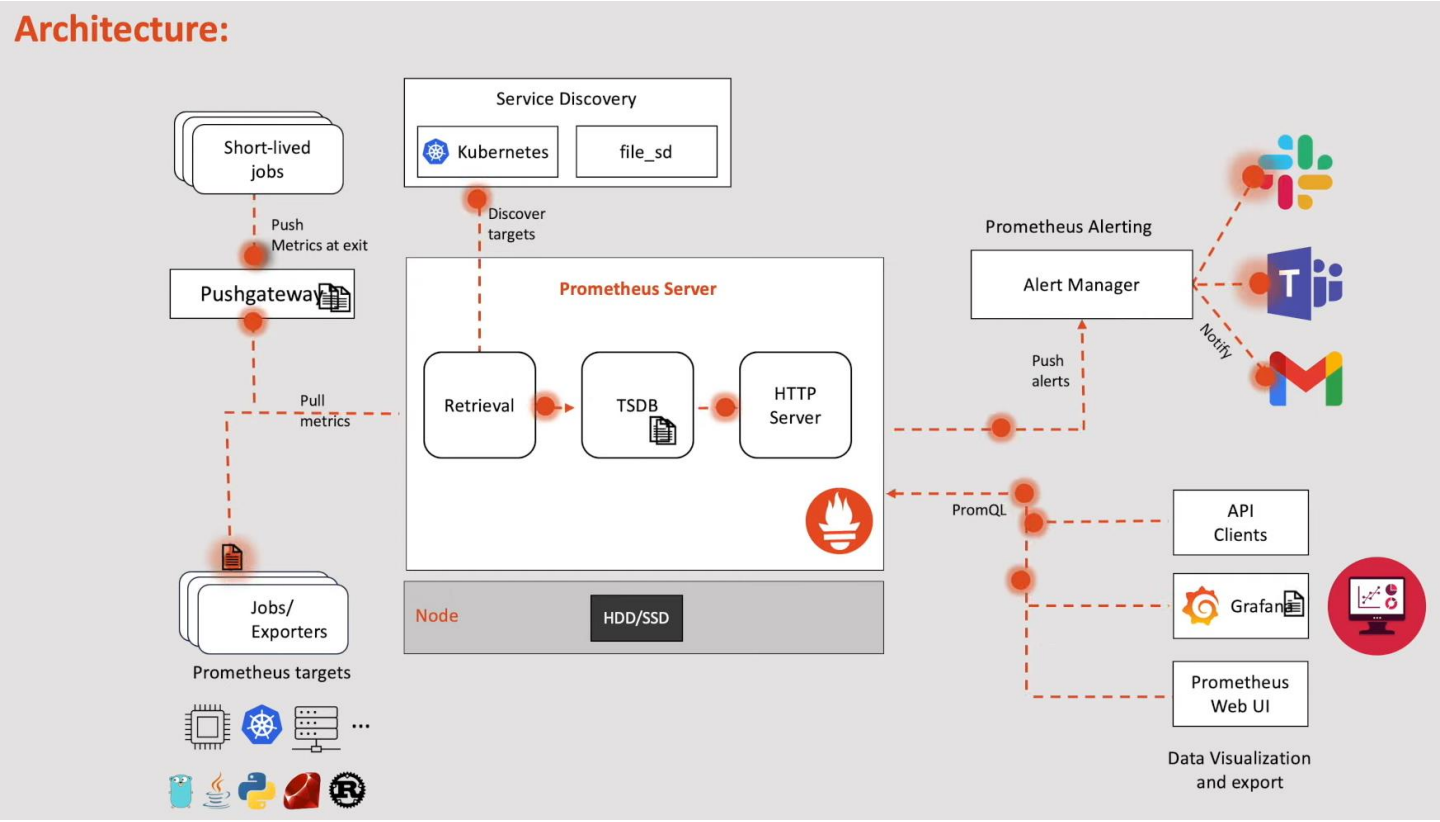
4. Facts About Prometheus

- **Open-Source**: Prometheus is an open-source tool widely adopted due to its community-driven nature.
- **Integration with Kubernetes**:
  - Prometheus easily integrates with Kubernetes, as many Kubernetes components expose metrics in the Prometheus format.
- **Comparison with Other Tools**:
  - While there are other monitoring solutions (e.g., Nagios, DataDog, New Relic), Prometheus is the **default standard** for Kubernetes monitoring.

Summary

- **Prometheus** is a powerful, open-source tool that enables the collection, storage, and analysis of time series data for observability and monitoring.
- It is widely used for tracking system metrics, setting up alerts, and gaining insights into application performance.

Architecture:



1. Prometheus Server

- **Core Component**: The **Prometheus Server** is the central part responsible for collecting and storing metrics.
- **Metrics Collection**: Prometheus collects metrics by scraping **HTTP endpoints** from various targets, such as:
  - Applications
  - Bare metal servers
  - Kubernetes clusters
  - Database instances

- **Time Series Database (TSDB)**:
    - Metrics are stored in a **local time series database**.
    - Default retention is **15 days**, but this can be configured or stored remotely (e.g., on S3).
- **API**: Prometheus provides an **API** to query the stored data, which can be accessed via **Grafana** or the **Prometheus UI**.

2. Client Libraries

- **Purpose**: Custom metrics for applications can be generated using **client libraries**.
- **Supported Languages**: Prometheus supports client libraries for various programming languages, such as **Go**, **Java**, **Python**, **Ruby**, and **Rust**.
- **Instrumentation**: Adding code to an application to expose its own metrics is called **instrumentation**. For example, tracking the number of to-do items in a to-do application.

3. Exporters

- **Purpose**: For systems that cannot be instrumented directly (like **Linux** or **MongoDB**), **exporters** help fetch existing metrics and expose them in Prometheus-compatible formats.
- **How It Works**:
    - Exporters fetch metrics from systems and expose them via HTTP endpoints.
    - Prometheus scrapes these endpoints to collect the data.
- **Examples of Exporters**:
    - **Node Exporter**: Fetches metrics from Linux systems.
    - **MongoDB Exporter**: Fetches metrics from MongoDB.

4. Push Gateway

- **Use Case**: For **short-lived applications** like batch jobs or Lambda functions that run briefly and exit, Prometheus's **pull-based system** doesn't work well.
- **Solution**: The **Push Gateway** allows these applications to **push** their metrics, and Prometheus can scrape the data from the Push Gateway.
- **Important Note**: The Push Gateway can introduce additional storage requirements, so it should be used mindfully.

5. Service Discovery

- **Purpose**: In a **containerized environment** (e.g., Kubernetes), targets like **pods** are frequently added and removed, making manual configuration of Prometheus for each new target cumbersome.
- **Solution**: Prometheus has **service discovery** that automatically detects and scrapes targets as they are added or removed.
- **Benefit**: Automated monitoring and scraping without manual intervention, which is critical in dynamic environments like Kubernetes.

6. PromQL and Prometheus UI

- **PromQL**: **Prometheus Query Language (PromQL)** is used to query and analyze the stored metrics.
- **Prometheus UI**:
    - **Access**: By default, accessible on port **1990** of the Prometheus server.
    - **Purpose**: Useful for ad-hoc queries and debugging.
    - **Limitations**: Not ideal for creating dashboards. For visualization, **Grafana** is recommended.

7. Alert Manager

- **Purpose**: Prometheus can trigger alerts based on specific conditions or thresholds.
- **How It Works**:
    - **Alert Rules**: Set conditions for alerts.
    - **Alert Manager**: When an alert condition is met, the Alert Manager sends notifications through various channels, such as **Slack**, **Email**, or **Microsoft Teams**.
- **Role**: The **Alert Manager** ensures Prometheus is also used for **alerting** alongside monitoring.

Summary

- **Prometheus** is an all-in-one tool for monitoring and alerting:
  - Collects metrics from configured targets at regular intervals.
  - Stores metrics in a time series database (TSDB).
  - Provides a UI and supports PromQL for querying metrics.
  - Sends notifications via the Alert Manager when defined thresholds are met.
- **Components**: Key components include the Prometheus Server, Client Libraries, Exporters, Push Gateway, Service Discovery, PromQL, and Alert Manager. Together, these components enable effective monitoring and alerting.

Prometheus Installation:

Installation using helm -> https://youtu.be/kmjfRm82Sms?si=KycIB2AitLmeo_f0