# 参考文档

官网： https://prometheus.io/docs/introduction/overview/

IBM（入门与实践）：

https://developer.ibm.com/zh/articles/cl-lo-prometheus-getting-started-and-practice/

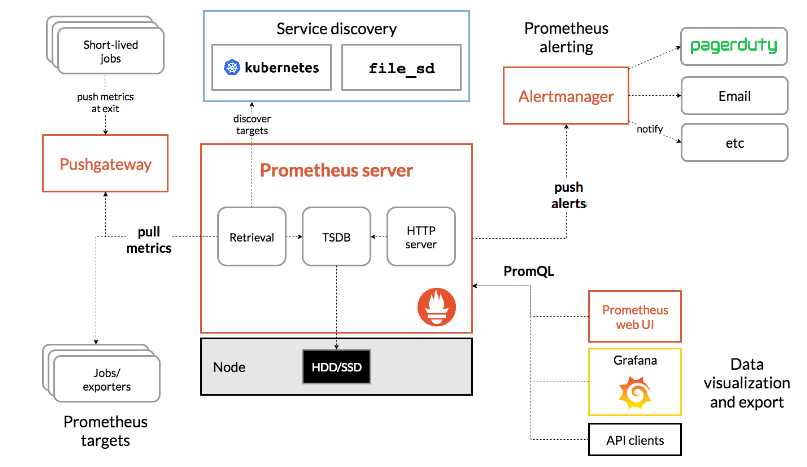
互联网常用系统监控工具对比：监控目标、方法、核心、工具、流程、指标（硬件、系统、应用）

https://blog.csdn.net/luoqinglong850102/article/details/106090904

Prometheus马哥亲讲 https://www.bilibili.com/video/BV1PT4y1P7bX?p=1

<https://www.bilibili.com/video/BV1mz4y1d7qw?from=search&seid=13974519123448433602>

# 架构



Prometheus 生态圈中包含了多个组件，其中许多组件是可选的：

* Prometheus Server: 用于收集和存储时间序列数据。
* Client Library: 客户端库，为需要监控的服务生成相应的 metrics 并暴露给 Prometheus server。当 Prometheus server 来 pull 时，直接返回实时状态的 metrics。
* Push Gateway: 主要用于\*\*短期\*\*的 jobs。由于这类 jobs 存在时间较短，可能在 Prometheus 来 pull 之前就消失了。为此，这次 jobs 可以直接向 Prometheus server 端推送它们的 metrics。这种方式主要用于服务层面的 metrics，对于机器层面的 metrices，需要使用 node exporter。
* Exporters: 用于暴露已有的第三方服务的 metrics 给 Prometheus。
* Alertmanager: 从 Prometheus server 端接收到 alerts 后，会进行去除重复数据，分组，并路由到对收的接受方式，发出报警。常见的接收方式有：电子邮件，pagerduty，OpsGenie, webhook 等。
* 一些其他的工具。

**大概的工作流程：**

1. Prometheus server 定期从配置好的 jobs 或者 exporters 中拉 metrics，或者接收来自 Pushgateway 发过来的 metrics，或者从其他的 Prometheus server 中拉 metrics。

2. Prometheus server 在本地存储收集到的 metrics，并运行已定义好的 alert.rules，记录新的时间序列或者向 Alertmanager 推送警报。

3. Alertmanager 根据配置文件，对接收到的警报进行处理，发出告警。

4. 在图形界面中，可视化采集数据。

# 基础入门

<https://developer.ibm.com/zh/articles/cl-lo-prometheus-getting-started-and-practice/>

16年加入CNCF， 成为受欢迎度仅次于 Kubernetes 的项目。

## 功能特点

* 强大的多维度数据模型：

1. 时间序列数据通过 metric 名和键值对来区分。

2. 所有的 metrics 都可以设置任意的多维标签。

3. 数据模型更随意，不需要刻意设置为以点分隔的字符串。

4. 可以对数据模型进行聚合，切割和切片操作。

5. 支持双精度浮点类型，标签可以设为全 unicode。

* 灵活而强大的查询语句（\*\*PromQL\*\*）：同一个查询语句，可以对多个 metrics 进行乘法、加法、连接、取分数位等操作。
* 易于管理： Prometheus server 是一个单独的二进制文件，可直接在本地工作，不依赖于分布式存储。
* 高效：\*\*平均每个采样点仅占 3.5 bytes\*\*，且一个 Prometheus server 可以处理数百万的 metrics。
* 使用 \*\*pull 模式\*\*采集时间序列数据，这样不仅有利于本机测试而且可以避免有问题的服务器推送坏的 metrics。
* 可以采用 \*\*push gateway\*\* 的方式把时间序列数据推送至 Prometheus server 端。
* 可以通过服务发现或者静态配置去获取监控的 targets。
* 有多种可视化图形界面。
* 易于伸缩。

## 基础知识

**1、监控系统组件**

- 指标数据采集（抓取）

- 指标数据存储

- 指标数据分析

- 告警

**2、监控体系**

* 系统级监控

- 系统监控：CPU/Load/Memory/Swap/Disk IO/Processes/Kernel Parameters

- 网络监控：网络设备、工作负载、网络延迟、丢包率

* 中间件及基础设备类系统监控

- 消息中间件：Kafka、RocketMQ、RabbitMQ

- Web服务容器：Tomcat、Jetty

- 数据库及缓存系统：Mysql、PostgreSQL、ElasticSearch、Redis

- 数据库连接池：

- 存储系统：Ceph等

* 应用层监控

- 衡量应用程序代码的状态和监控

* 业务层监控

- 衡量应用程序的价值，例如电子商务网站的销售量

- QPS、DAU日活、转化率

- 业务接口：登录数、注册数、订单量、搜索量、支付量

**3、云原生时代的可观测性**

可观测性系统：

- 指标监控Metrics：随时间推移产生的一些与监控相关的可聚合数据点

- 日志监控Logging：离散式日志和事件

- 链路追踪Tracing：分布式应用调用链跟踪

CNCF将可观测性和数据分析归类一个单独类别，并划分四个子类

- 监控系统：Prometheus

- 日志系统：ElasticStack和PLG Stack

- 分布式调用链跟踪系统：Zipkin、Jaeger、SkyWalking、Pinpoint

- 混沌工程系统：ChaoMonkey、ChaosBlade

**4、监控方法论**

1) Google四个黄金指标

用于服务级别帮助衡量终端用户体验、服务终端、业务影响等层面问题

​ 适用于应用及服务监控

- 延迟Latency：如HTTP请求平均延迟，需区分请求和失败

- 流量Traffic：每秒处理HTTP请求数或数据库事务数量

- 错误Errors：请求失败速率

- 饱和度Saturation：衡量资源使用情况，如内存、CPU、I/O、磁盘使用量

2) Netflix的USE方法

Utilization Saturation and Errors Method

​ 分析系统性能问题，指导用户快速识别资源瓶颈及错误方法

​ 应用于主机指标监控

3) Weave Cloud的RED方法

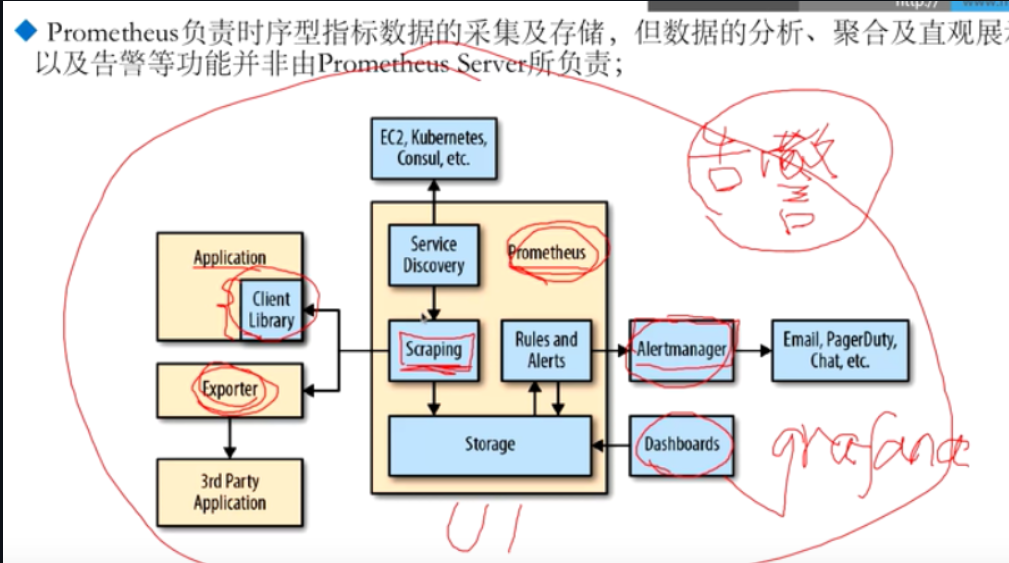
​ 基于Google四个黄金指标原则下结合Prometheus及Kubernetes容器实践，细化和总结的方法论，特别适合云原生应用及微服务架构应用的监控和度量

- Request Rate：每秒接收请求数

- Request Errors：每秒失败请求数

- Request Duration：每个请求花费时长

# Prometheus介绍



内置的UI仅用于PromSQL的调试，具体展示还是依赖Grafana组件

内部可以产生告警，但是如何通知展示，依赖AlertManager组件

内部提供了PromQL（查询语言）

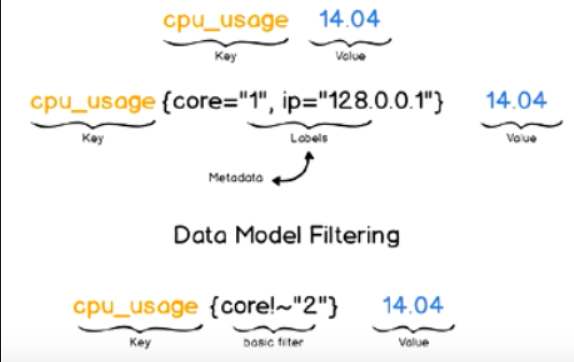
**Prometheus数据模型**

仅用于以键值方式存储时序式的聚合数据，并不支持存储文本

- 键：称为指标Metrics，常意味着CPU速率、内存使用率或分区空闲比例等

- 同一指标可能适配到多个目标或设备， 因而它使用标签作为元数据，从而为Metrics 添加更多信息描述维度（截图第2行，表示IP等信息）

- 标签还可作为过滤器进行指标过滤及聚合运算（截图第3行中间为过滤条件）



prometheus使用4种方法来保存监视的指标：

* Counter计数器：保存单调递增型数据，如站点访问次数，不能为负数，也不支持减少，可以重置回0；
* Gauge仪表盘：存储有起伏特征的指标数据，如内存空闲大小；Gauge式Counter的超集，但存在指标数据丢失的可能性（Counter能让用户确切了解指标随时间变化状态，Gauge可能随时间流逝而精准度越来越低
* Histogram直方图，会在一段时间方位内对数据进行采样，并将其计入可配置的bucket中；Histogram能存储更多信息，包括样本值分布在每个bucket中数量、所有样本值之和及总的样本数量，从而Prometheus能使用内置的函数进行如下操作：

- 计算样本平均值：以值得总和除以值得数量

- 计算样本分位置：分位数有助于了解符合特定标准的数据个数；如评估响应时长超过 1s的请求比例，若超过20%即发送告警

* Summary摘要：Histogram扩展类型，但它是直接由被检测端自行聚合计算出分位数，并将计算结果响应给Prometheus Server的样本采集请求。因而，分位数计算由监控端完成。

<https://www.bilibili.com/video/BV1PT4y1P7bX?p=2&spm_id_from=pageDriver>

# 适合和不适合场景

## 不适合场景

由于数据采集可能会有丢失，所以 Prometheus 不适用对采集数据要 100% 准确的情形。但如果用于记录时间序列数据，Prometheus 具有很大的查询优势，此外，Prometheus 适用于微服务的体系架构。 If you need 100% accuracy, such as for per-request billing, Prometheus is not a good choice as the collected data will likely not be detailed and complete enough.

## 适合场景

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. In a world of microservices, its support for multi-dimensional data collection and querying is a particular strength.

# 配置文件格式

