## 特点：

1. 数据精细，精确到1-5秒
2. 部署速度快
3. 周边插件很丰富如：Exporter、PushgateWay
4. 大量使用的函数，实现复杂逻辑的监控（弯曲凸起），数学函数，也就是PromQL
5. 可以嵌入很多开源工具内部进行监控：Nginx、数据库

**缺点**：

1. 精度高，集群数量大
2. 磁盘资源耗费大

**各种级别的监控**

1. 业务级别监控：用户访问量QPS,DAU日活，访问状态（http请求）
2. 系统级别监控：CPU/内存/硬盘/IO/TCP
3. 网络监控：托管服务器（公有云），连通性、丢包率、公有与上可用区（不同可用区创建虚拟机）
4. 日志监控：往往单独设计和搭建，全部种类日志都需要采集

**采集数据的意义**

对于传统的监控数据采集方式，例如CPU的其中状态参数、采集用户每秒访问的请求量，这些基本单位，对于采集回来的基本单位数据，没有形成监控数据或者报警阈值，则采集的数据没有意义

传统的监控系统：得到报警，但是是否有利于我们通过监控找到问题？

**监控系统的未来**

**完善自愈式监控体系**：监控和报警，总归只能发现问题，出现问题之后依然需要人工干预，未来打造一种自愈系统，各种层级问题，都会被各种自动化、持续集成、人工智能、灾备、统缓冲等技术自行修复。

**真实链路式监控**：监控和报警的准确性、真实性发展到一个理想化模型——

例如：当系统发出报警信息，往往是各层级的报警，一大堆一起发出，报警的真实引起的问题的地方被掩盖住了非常不利于我们及时发现问题、处理问题。

如：一个数据库的联合查询，对系统的资源消耗太大，造成各方面的资源大量的消耗，引起各种链路问题，于是乎各层级的报警接踵而至，日志报警、慢查询报警、CPU报警，企业级监控用户流量下降报警，一堆问题全部暴露，真正原因DB查询反而没有发现。无关报警，全部忽略。（精确、真实：因为其提供的是底层的数据监控）

**Prometheus的系统架构**

**3种常用的metric类型**

**使用及函数**