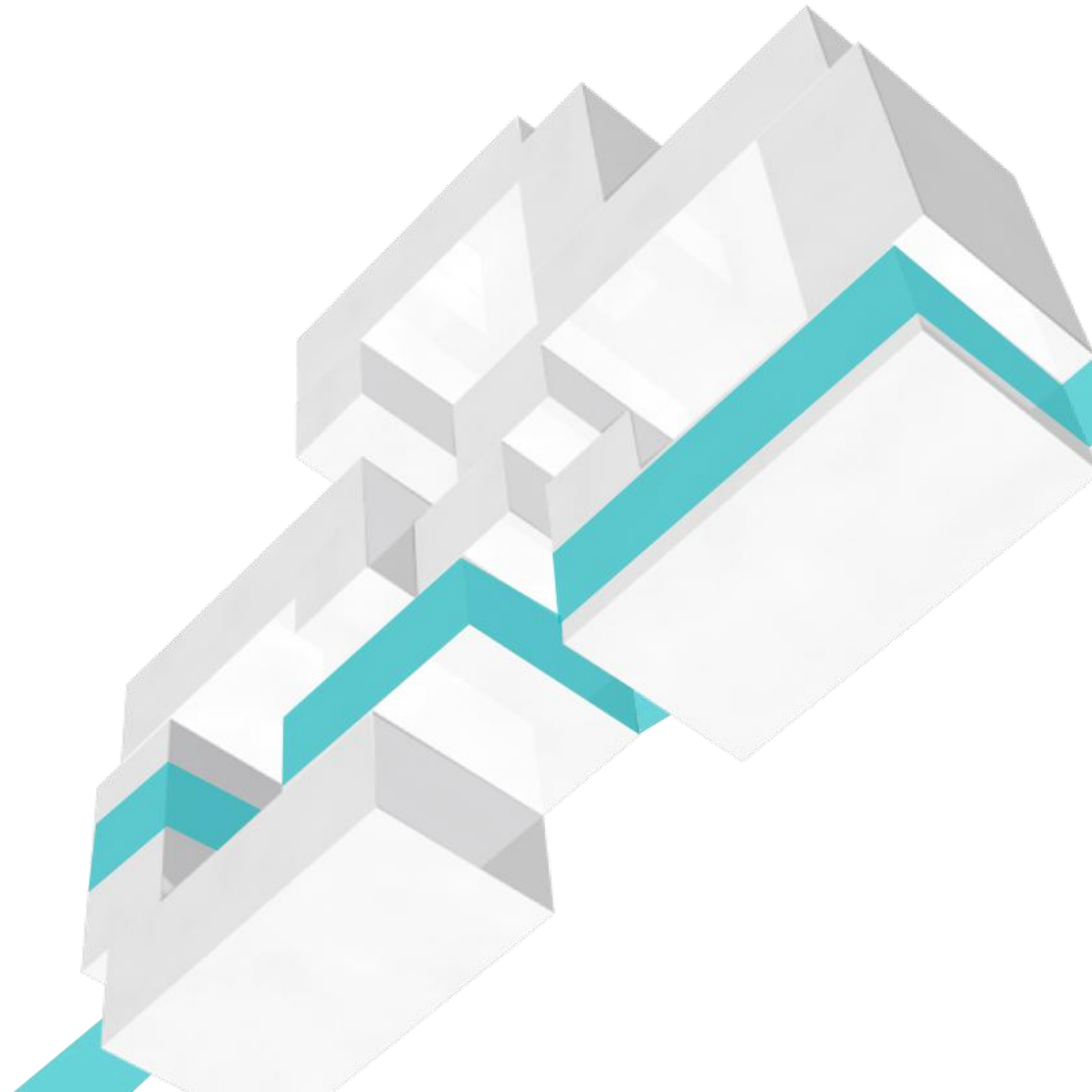
云智慧
Cloudwise

云原生时代监控分享

云原生时代热门监控利器解读分析



演讲人：服务效能部

日期：2022/03/31

目录

PART 01

监控简介

监控的目的

监控的目的

建立完善的监控体系可达到以下目的：

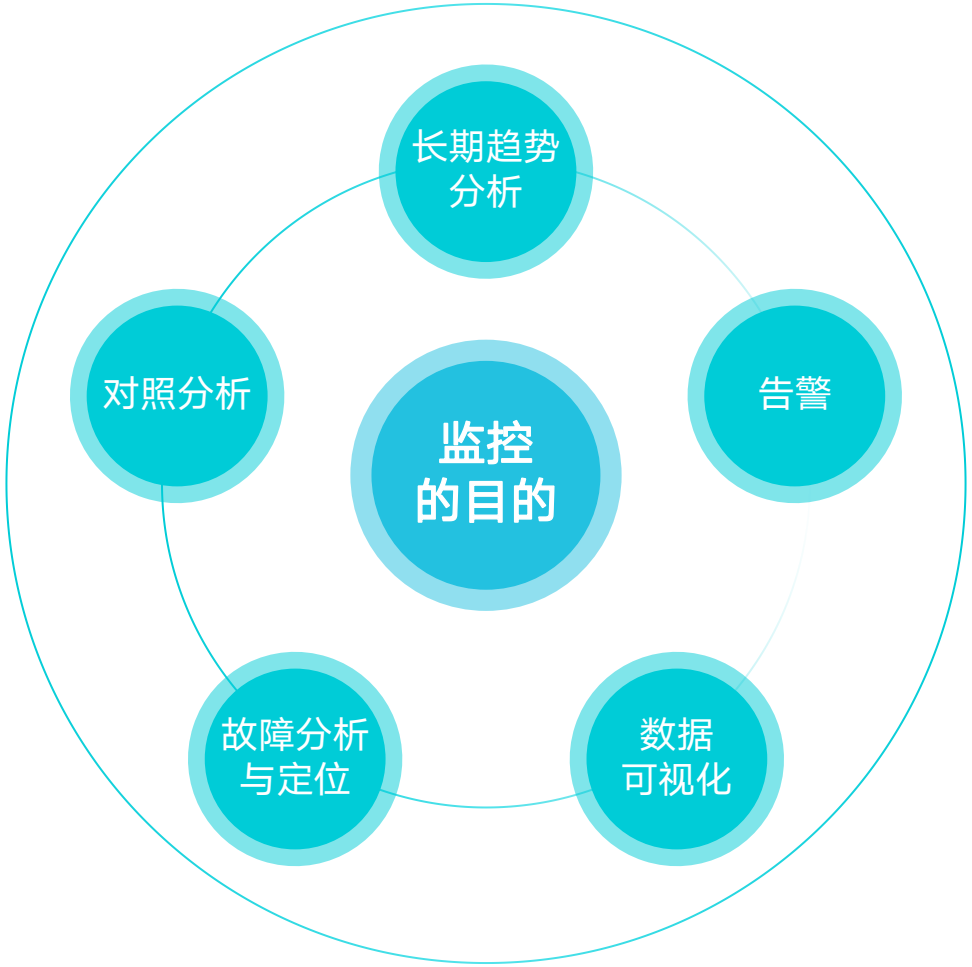
通过对监控样本数据的持续收集和统计，对监控指标进行长期趋势分析。

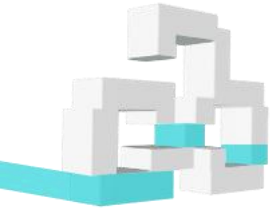
两个版本的系统运行资源使用情况的差异如何？在不同容量情况下系统的并发和负载变化如何？通过监控能够方便的对系统进行跟踪和比较。

当系统出现或者即将出现故障时，监控系统需要迅速反应并通知管理员，从而能够对问题进行快速的处理或者提前预防问题的发生，避免出现对业务的影响。

当问题发生后，需要对问题进行调查和处理。通过对不同监控监控以及历史数据的分析，能够找到并解决根源问题。

通过可视化仪表盘能够直接获取系统的运行状态、资源使用情况、以及服务运行状态等直观的信息。





监控维度

级别	监控内容
网络	网络协议：http、dns、tcp、icmp； 网络硬件：路由器，交换机等
主机	资源用量（CPU、MEM、Storage）
容器	资源用量（CPU、MEM、Storage）
应用(包括Library)	延迟，错误，QPS，内部状态等
中间件状态	资源用量，以及服务状态
编排工具	集群资源用量，调度等

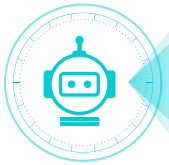
四个监控黄金指标

延迟：服务请求所需时间。

记录用户所有请求所需的时间，重点是要区分成功请求的延迟时间和失败请求的延迟时间。

通讯量：监控当前系统的流量，用于衡量服务的容量需求。

流量对于不同类型的系统而言可能代表不同的含义。例如，在HTTP REST API中，流量通常是每秒HTTP请求数；



错误：监控当前系统所有发生的错误请求，衡量当前系统错误发生的速率。

对于失败而言有些是显式的(比如，HTTP 500错误)，而有些是隐式(比如，HTTP响应200，但实际业务流程依然是失败的)。对于一些系统内部的异常，则可能需要直接从服务中添加钩子统计并进行获取。

饱和度：衡量当前服务的饱和度。

主要强调最能影响服务状态的受限制的资源。 因为通常情况下，当这些资源达到饱和后，服务的性能会明显下降。同时还可以利用饱和度对系统做出预测。比如，“磁盘是否可能在4个小时候就满了”。

PART 02

Prometheus监控系统

Prometheus的优势

多平台部署兼容性好，可以处理数以百万的监控指标，每秒处理数十万的数据点。

对kubernets（趋势）支持友好，随着Kubernets在容器调度和管理上确定领头羊的地位，Prometheus也成为Kubernets容器监控的标配。

使用Prometheus可以快速搭建监控服务，并且可以非常方便地在应用程序中进行集成。
采集客户端丰富（官方，第三方，自定义）

Prometheus内置了一个强大的数据查询语言PromQL。 通过PromQL可以实现对监控数据的查询、聚合。同时PromQL也被应用于数据可视化(如Grafana)以及告警当中。

可功能分区(sharding)+联邦集(federation)可以对其进行扩展

目前已支持Kubernets、Consul，Etcd等，可以减少运维人员手动配置的工作量（在容器环境尤为重要）

Prometheus架构

Prometheus Server

Retrieval通过 HTTP Server定时向服务动态发现的目标抓取metrics（指标）数据，每个抓取目标都需要暴露一个HTTP服务接口（符合Prometheus规范）用于Prometheus定时抓取。并将采集来的监控数据持久化后存在TSDB中

PromQL查询

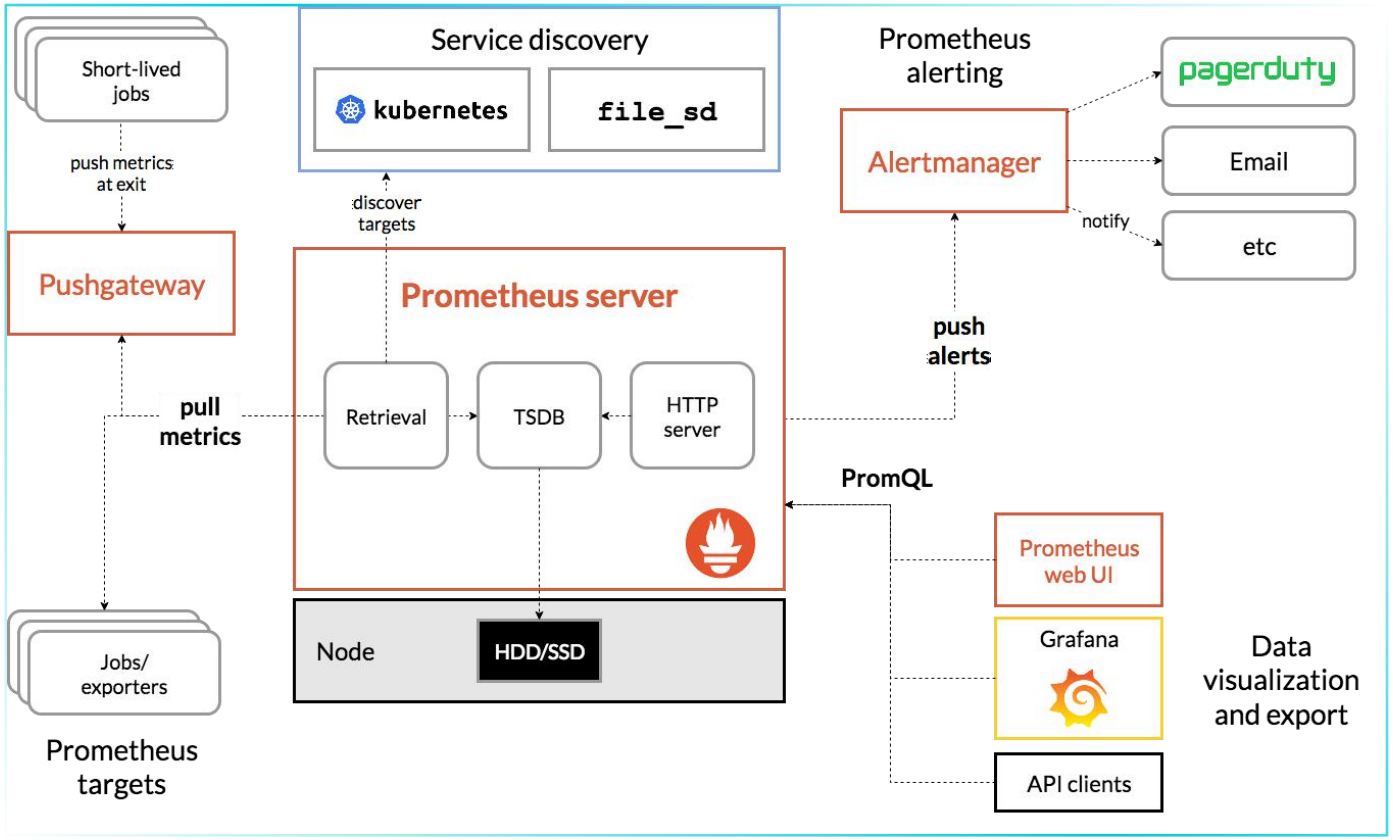
可以通过Prometheus WebUi，Api Clients或Grafana使用PromQL来查询各种指标数据

告警推送

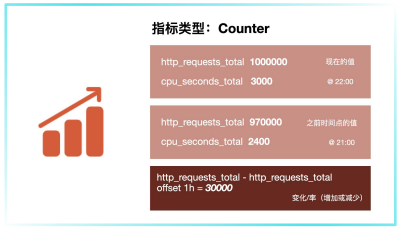
将计算后超过阈值的告警发送至Alertmanager，经过Alertmanager的进一步分组，抑制，静默后发送到更丰富的告警通道。

采集目标

被采集的目标既可以是各种官方Exporter，也可以是实现了Prometheus规范的三方接口（如Nacos和Arrangodb的metrics接口），或者PushGateway这种

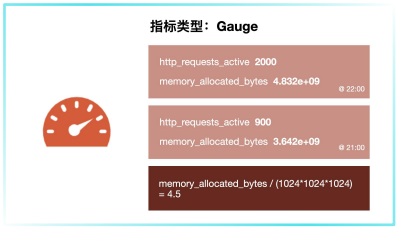


指标类型



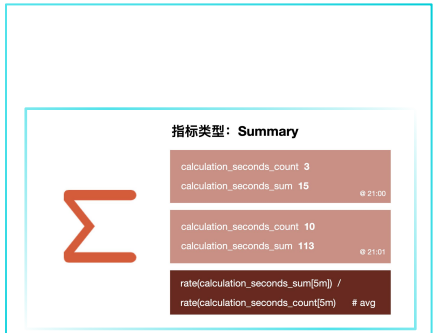
Counter

只增不减的计数器，对于存储诸如服务的 HTTP 请求数量或使用的 CPU 时间之类的信息非常有用。



Gauge

可增可减的仪表盘，该指标侧重于反应系统的当前状态



Summary

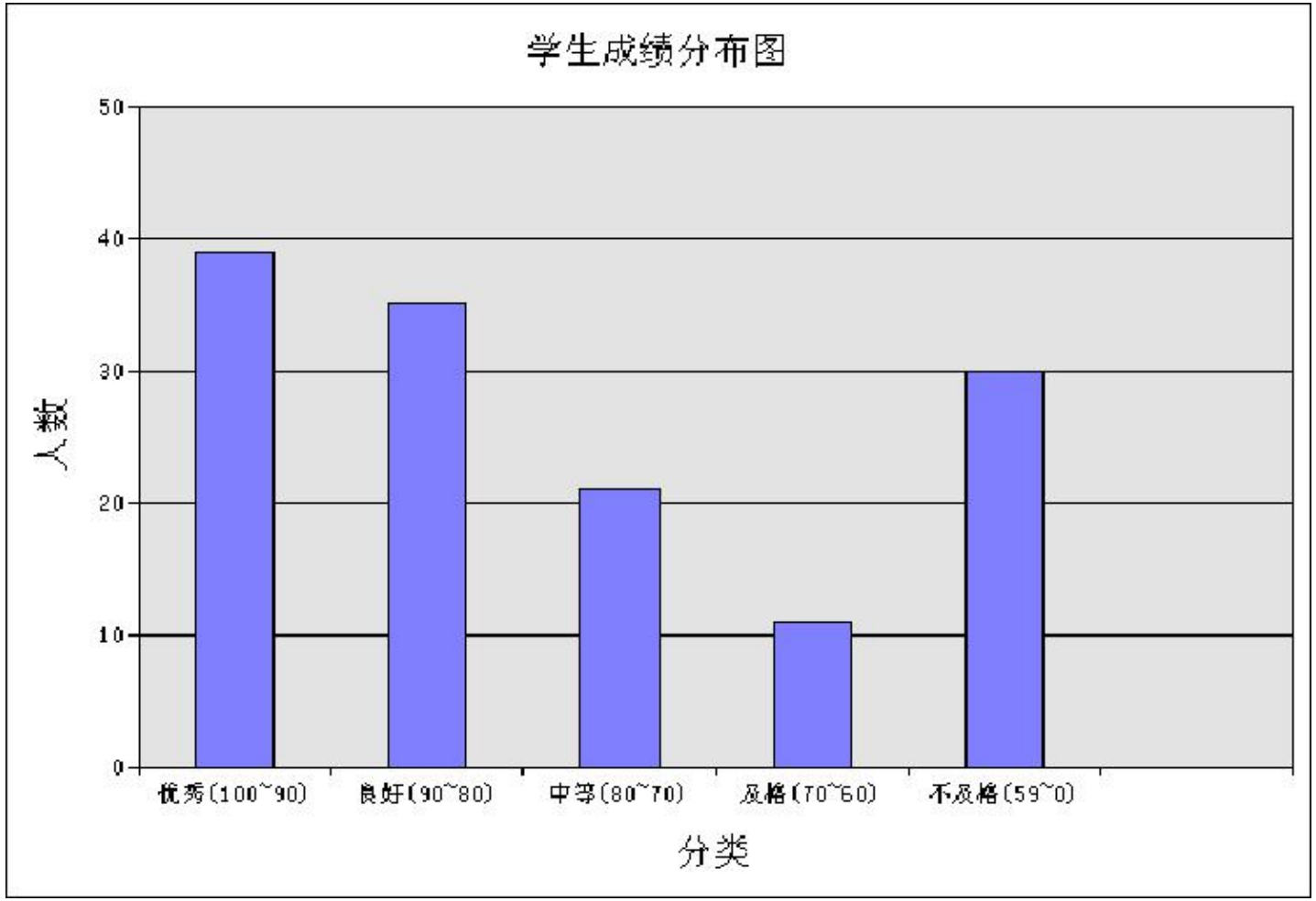
摘要用于记录某些东西的平均大小，可能是计算所需的时间或处理的文件大小，摘要显示两个相关的信息：count（事件发生的次数）和 sum（所有事件的总大小）

Histogram

Histogram 指标直接反应了在不同区间内样本的个数，区间通过标签 le 进行定义。

Histograms概念

Histograms 被叫主直方图或者柱状图, 主要用于统计指标值的一个分布情况



- Bucket: 设置横轴区间, 只设置上限 不设下限

- 0 ~ 100
- 0 ~ 90
- 0 ~ 80
- 0 ~ 70
- 0 ~ 60

Histograms典型应用场景:
统计请求耗时分布

- 0 ~ 100ms 请求个数
- 0 ~ 500ms 请求个数
- 0 ~ 5000ms 请求个数

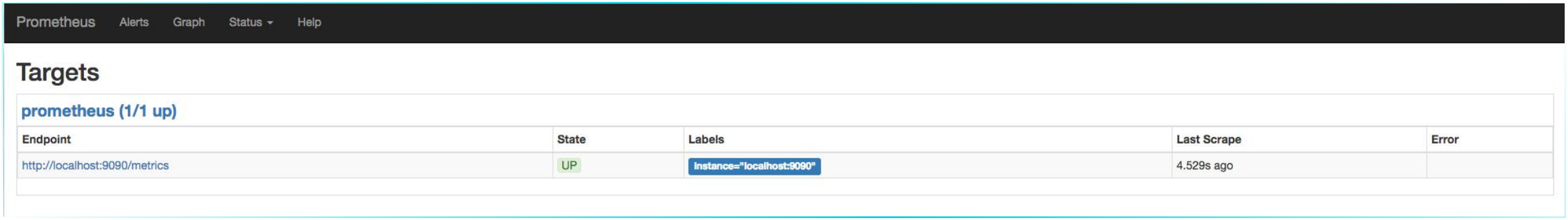
常用Exporter

范围	常用Exporter
数据库	MySQL Exporter, Redis Exporter, MongoDB Exporter, MSSQL Exporter等
硬件	Apcupsd Exporter，IoT Edison Exporter，IPMI Exporter, Node Exporter等
消息队列	Beanstalkd Exporter, Kafka Exporter, NSQ Exporter, RabbitMQ Exporter等
存储	Ceph Exporter, Gluster Exporter, HDFS Exporter, ScaleIO Exporter等
HTTP服务	Apache Exporter, HAProxy Exporter, Nginx Exporter等
API服务	AWS ECS Exporter， Docker Cloud Exporter, Docker Hub Exporter, GitHub Exporter等
日志	Fluentd Exporter, Grok Exporter等
监控系统	Collectd Exporter, Graphite Exporter, InfluxDB Exporter, Nagios Exporter, SNMP Exporter等
其它	Blockbox Exporter, JIRA Exporter, Jenkins Exporter， Confluence Exporter等

任务和实例

通过在prometheus.yml配置文件中，添加如下配置。

```
scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']
```



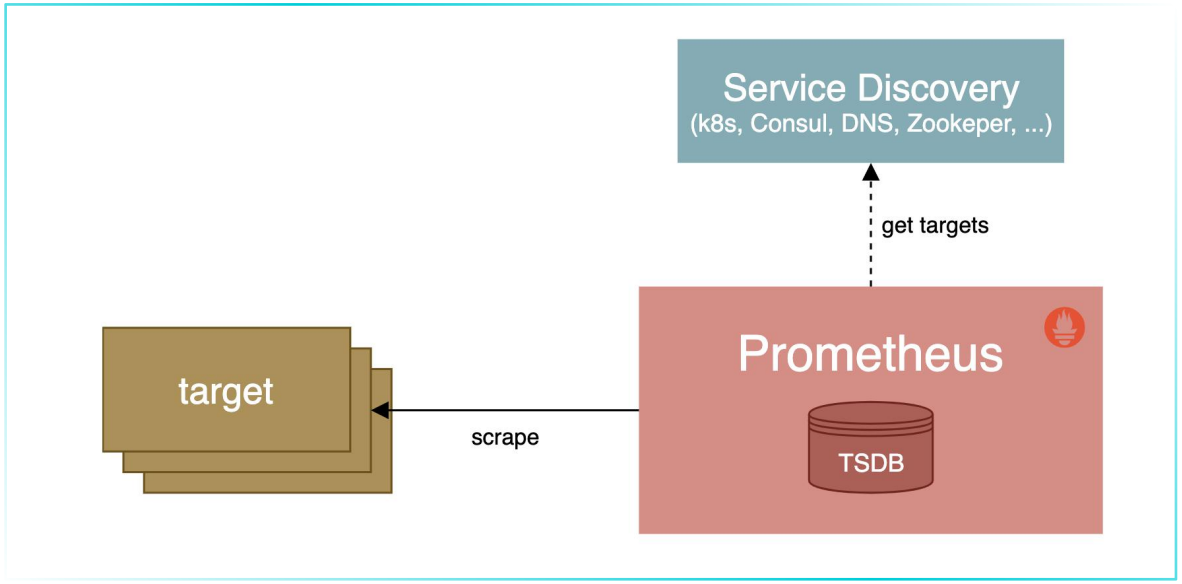
采集目标

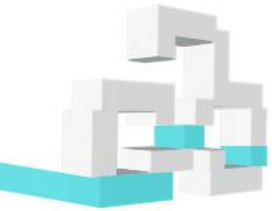
服务发现

Prometheus 已经支持多种内置的服务发现机制

我们都可以通过 Prometheus 配置文件中的 scrape_config 部分进行配置，Prometheus 会不断更新动态的抓取目标列表，自动停止抓取旧的实例，开始抓取新的实例，Prometheus 特别适合运行于 Kubernetes 集群下面，可以自动发现监控目标。

在 Kubernetes 下，Promethues 通过与 Kubernetes API 集成，主要支持 5 中服务发现模式，分别是：Node、Service、Pod、Endpoints、Ingress。





Kubernetes集群监控方案

- cAdvisor：cAdvisor 是 Google 开源的容器资源监控和性能分析工具，它是专门为容器而生，本身也支持 Docker 容器。目前已经内置在kubelet组件中。

- kube-state-metrics：kube-state-metrics 通过监听 API Server 生成有关资源对象的状态指标，比如 Deployment、Node、Pod，需要注意的是 kube-state-metrics 只是简单提供一个 metrics 数据，并不会存储这些指标数据，所以我们可以使用 Prometheus 来抓取这些数据然后存储。主要关注的是业务相关的一些元数据，比如 Deployment、Pod、副本状态等

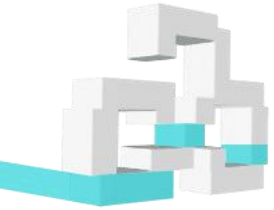
- metrics-server：metrics-server 也是一个集群范围内的资源数据聚合工具，是 Heapster 的替代品，同样的，metrics-server 也只是显示数据，并不提供数据存储服务。主要关注的是资源度量 API 的实现，比如 CPU、文件描述符、内存、请求延时等指标。

PART 03

Grafana数据可视化

数据面板


New dashboard
Last 6 hours
CPU使用率
7
6
5
4
3
2
1
12:00
13:00
14:00
15:00
16:00
17:00
node1:9100
node2:9100
Legend



数据面板

类型	常用内容
Time series	基于时间的折线图、面积图和条形图的数据可视化
Bar Chart	数据分类的可视化
Stat	用于数据统计相关的数据数据可视化
Gauge	显示与最小值和最大值相关的当前值的可视化
Table	表格布局中显示数据内容
Pie Chart	饼状图面板可视化
Heatmap	显示值分布

https://grafana.com/grafana/plugins/?type=panel

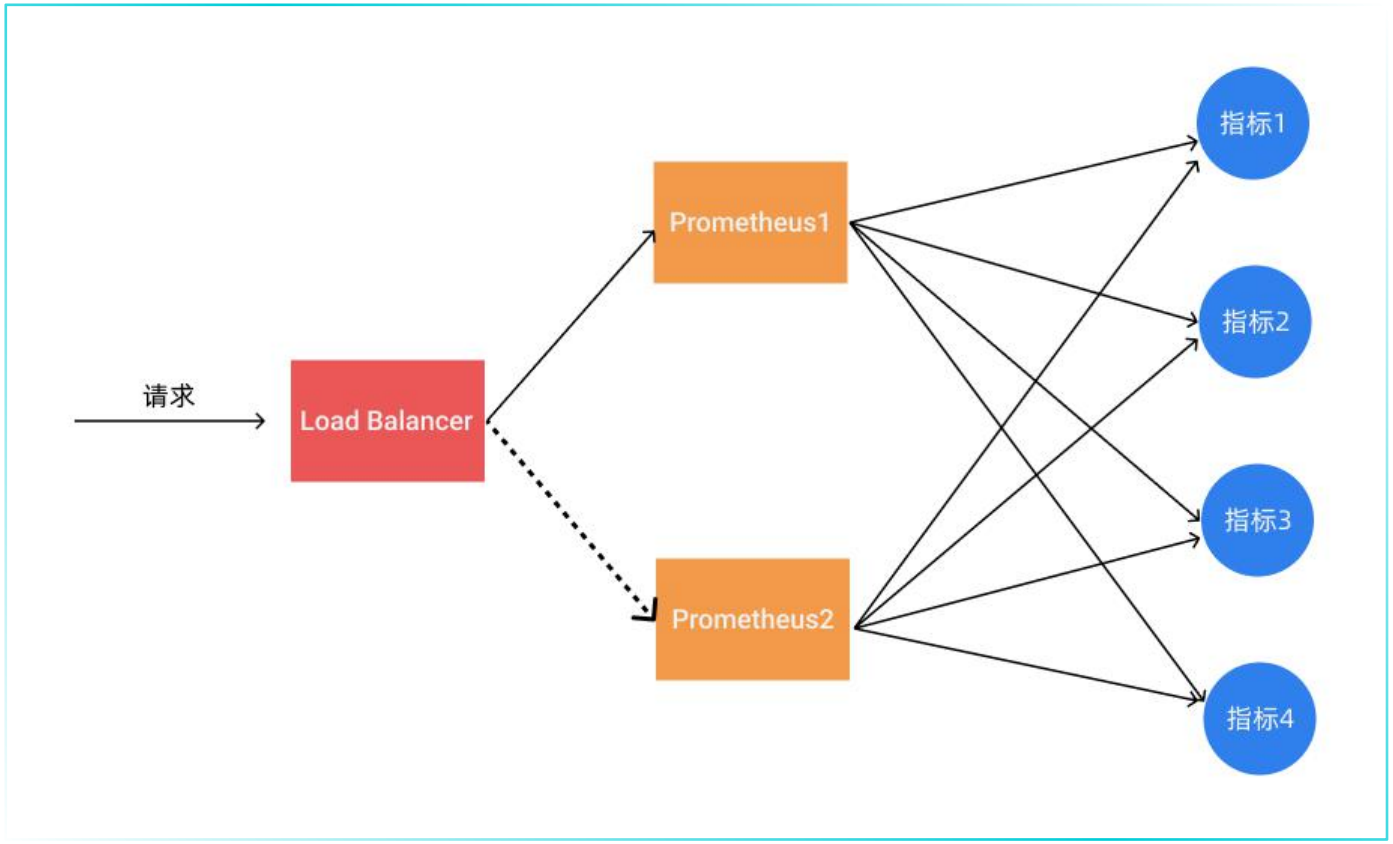
PART 04

Prometheus高可用

Prometheus高可用

可用性场景

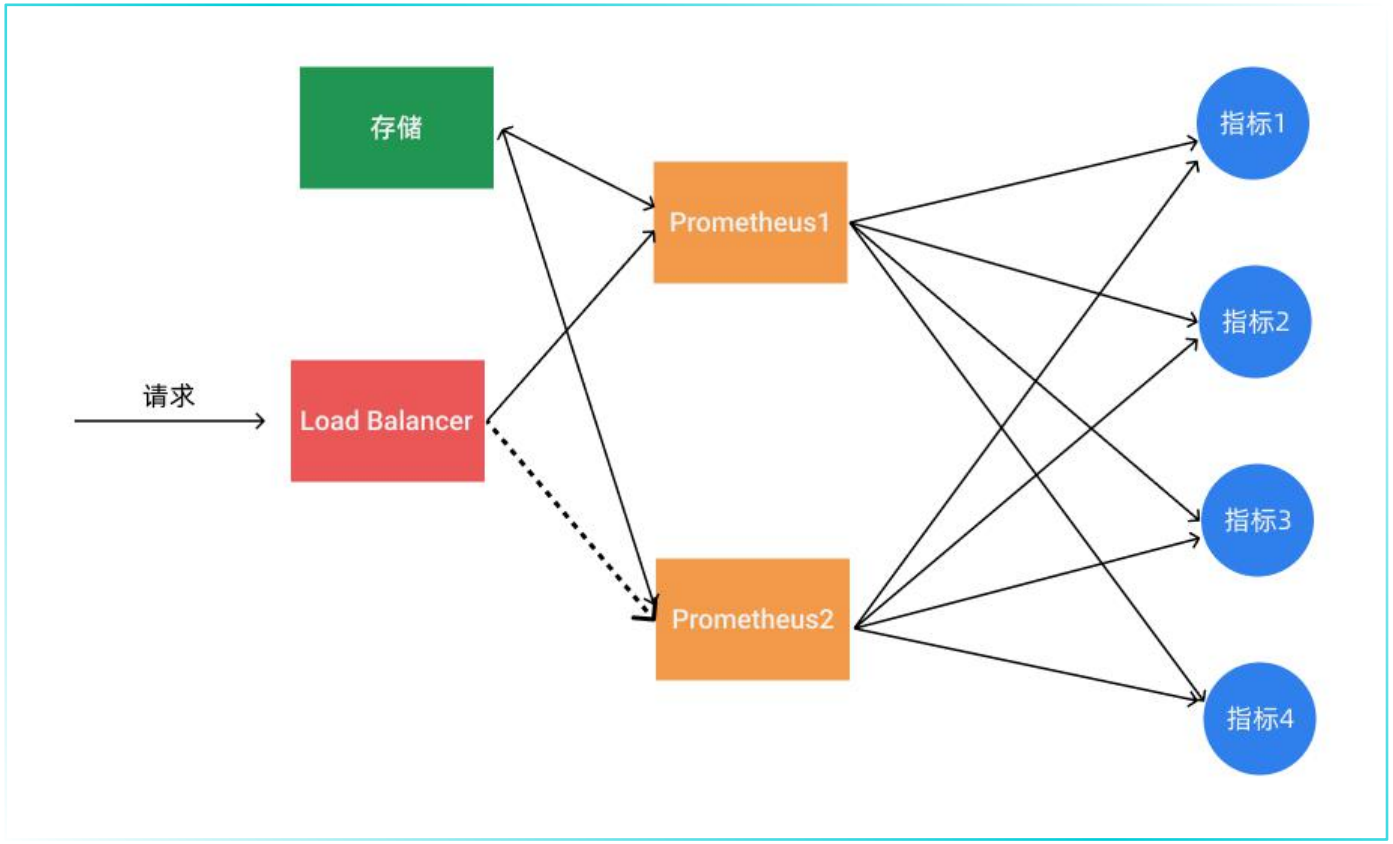
这个方式来满足服务的可用性应该是平时我们使用得最多的一种方式，当一个实例挂掉后从 LB 里面自动剔除掉，而且还有负载均衡的作用，可以降低一个 Prometheus 的压力，但这种模式缺点也是非常明显的，就是不满足数据一致性以及持久化问题，因为 Prometheus 是 Pull 的方式，即使多个实例抓取的是相同的监控指标，也不能保证抓取过来的值就是一致的，更何况在实际的使用过程中还会遇到一些网络延迟问题，所以会造成数据不一致的问题，不过对于监控报警这个场景来说，一般也不会要求数据强一致性，所以这种方式从业务上来说是可以接受的，因为这种数据不一致性影响基本上没什么影响。这种场景适合监控规模不大，只需要保存短周期监控数据的场景。



Prometheus高可用

数据持久化场景

在给 Prometheus 配置上远程存储过后，我们就不用担心数据丢失的问题了，即使当一个 Prometheus 实例宕机或者数据丢失过后，也可以通过远程存储的数据进行恢复。

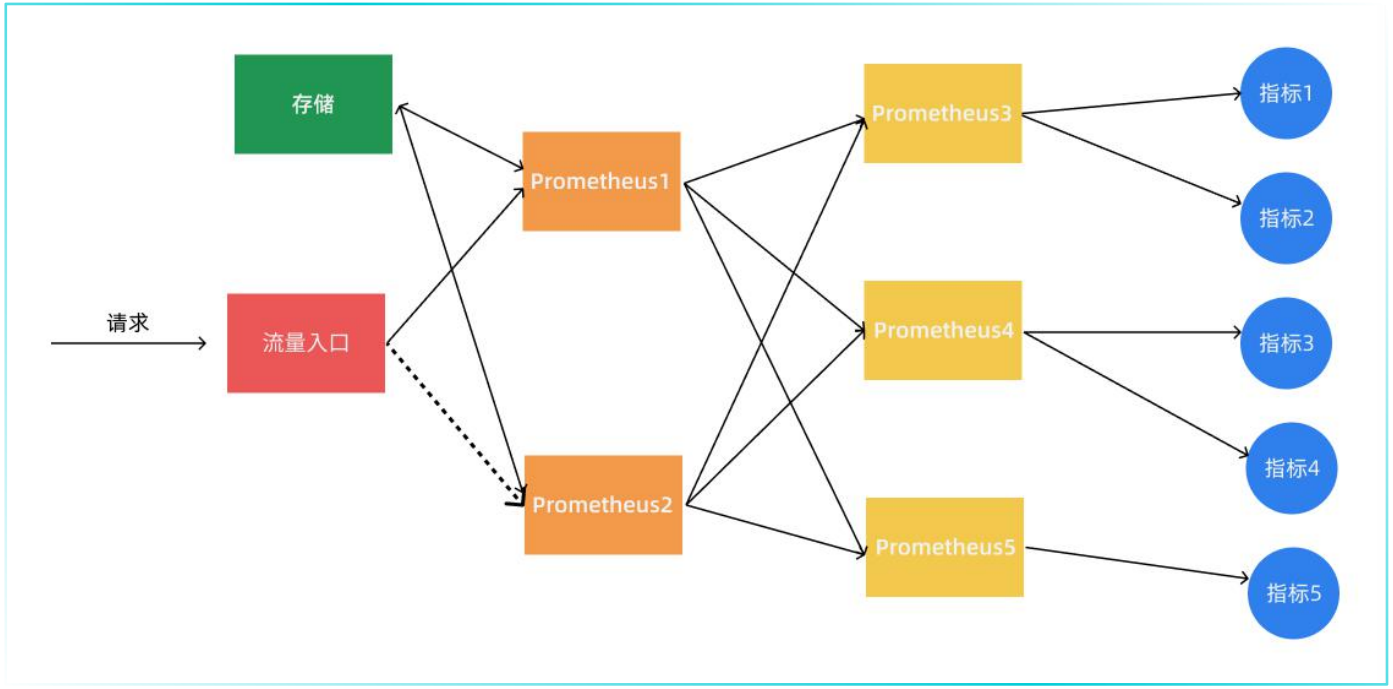


Prometheus高可用

联邦集群

当单个 Promthues 实例无法处理大量的采集任务时，这个时候我们就可以使用基于 Prometheus 联邦集群的方式来将监控任务划分到不同的 Prometheus 实例中去。

我们可以将不同类型的采集任务划分到不同的 Prometheus 实例中去执行，进行功能分片，比如一个 Prometheus 负责采集节点的指标数据，另外一个 Prometheus 负责采集应用业务相关的监控指标数据，最后在上层通过一个 Prometheus 对数据进行汇总。



Tips

当单个采集任务的目标数量过大时，可以考虑在实例级别进行功能分区。将一个任务的不同实例的监控数据采集任务划分到不同的Prometheus实例。

Prometheus高可用

Sidecar

连接 Prometheus，并把 Prometheus 暴露给查询网关（Querier/Query），以供实时查询，并且可以上传 Prometheus 数据给云存储，以供长期保存

Querier/Query

实现了 Prometheus API，与汇集底层组件（如边车组件 Sidecar，或是存储网关 Store Gateway）的数据

Store Gateway

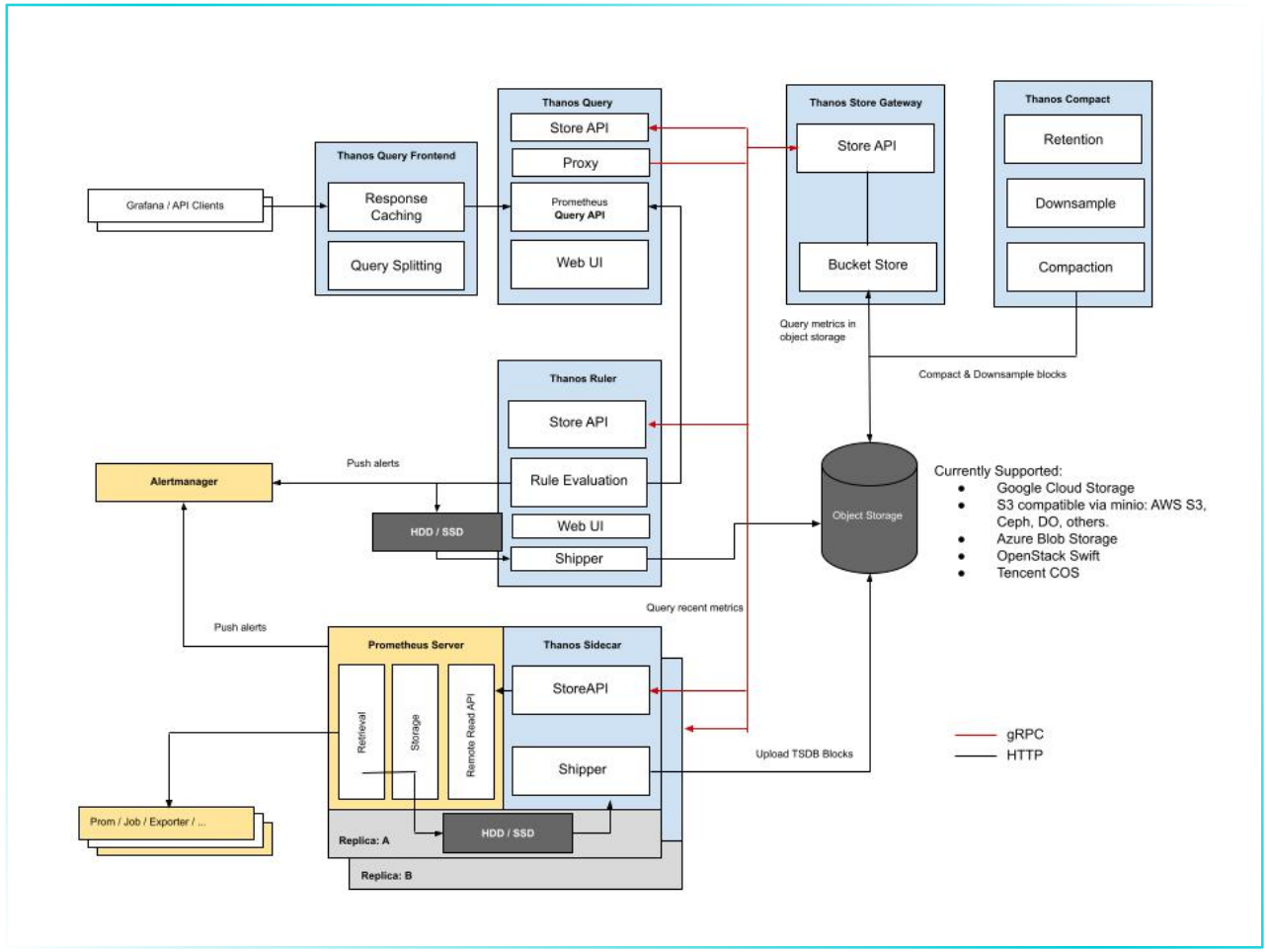
将云存储中的数据内容暴露出来

Compactor

将云存储中的数据进行压缩和下采样

Ruler

针对监控数据进行评估和报警



Prometheus高可用

Querier/Query

实现了 Prometheus API，与汇集底层组件（如边车组件 Sidecar，或是存储网关 Store Gateway）的数据

Store Gateway

将云存储中的数据内容暴露出来

Compactor

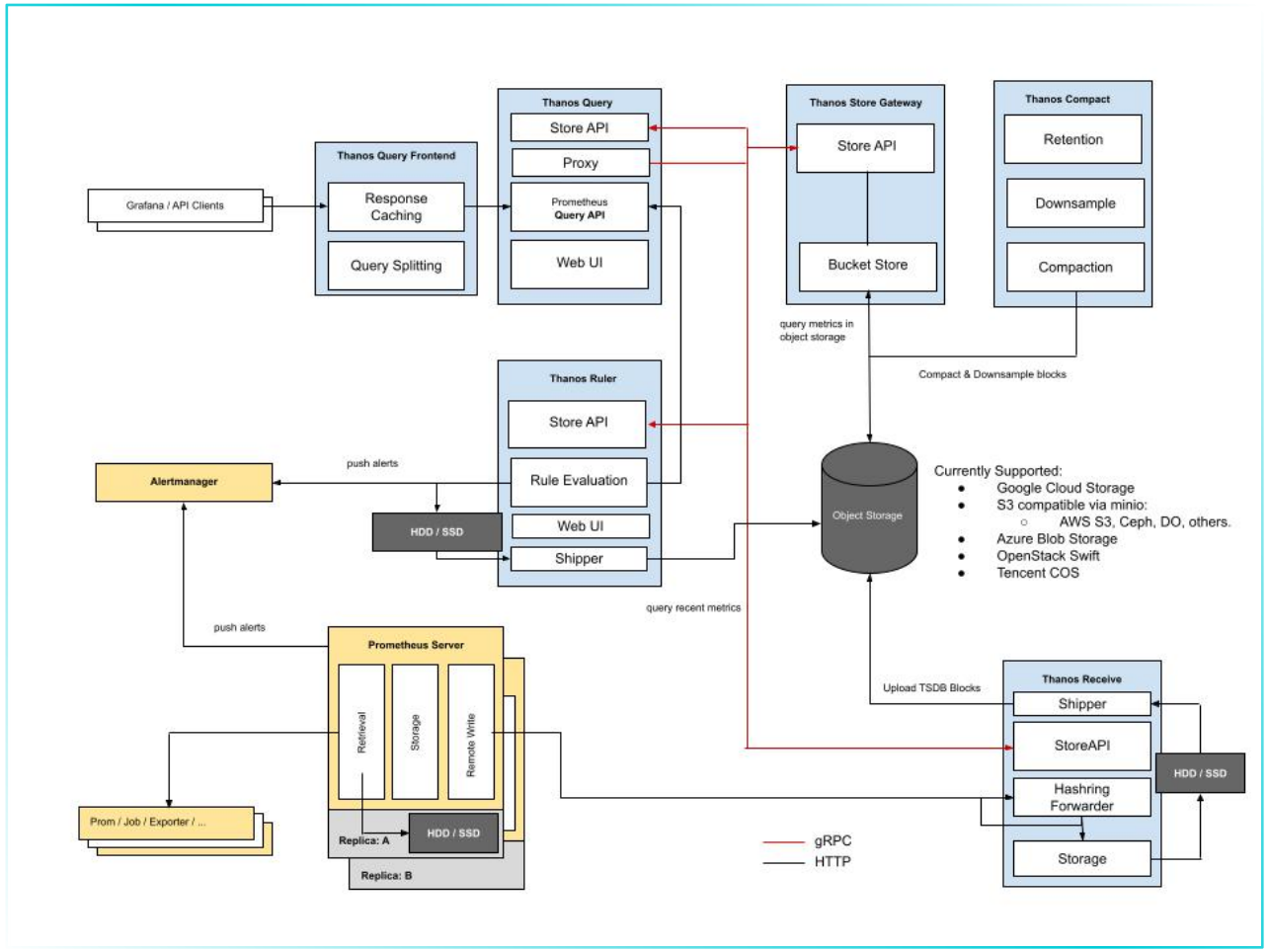
将云存储中的数据进行压缩和下采样

Receiver

从 Prometheus 的 remote-write WAL（Prometheus 远程预写式日志）获取数据，暴露出去或者上传到云存储

Ruler

针对监控数据进行评估和报警



Thanos的优势

以 Querier 作为统一的查询入口，其自身实现了 Prometheus 的查询接口和 StoreAPI，可为其他的 Querier 提供查询服务，在查询时会从每个 Prometheus 实例的 Sidecar 和 Store Gateway 获取到指标数据。

每个数据块都会带有特定的集群标签，Querier 在做查询时会去除集群标签，将指标名称和标签一致的序列根据时间排序合并。虽然指标数据来自不同的采集源，但是只会响应一份结果而不是多份重复的结果。

每个 Prometheus 本身不存储长时间的数据，Sidecar 会将 Prometheus 已经持久化的数据块上传到对象存储中。Compactor 会定时将远端对象存储中的长期数据进行压缩，并且根据采样时长做清理，节约存储空间。

Querier 是无状态服务，天生支持水平拓展和高可用。Store、Rule 和 Sidecar 是有状态服务，在多副本部署的情况下也支持高可用，不过会产生数据冗余，需要牺牲存储空间。

需要合并多个集群的查询结果时，仅需要在每个集群的 Querier 之上再添加一层 Querier 即可，这样的层层嵌套，可以使得集群规模无限制拓展。

当 Prometheus 的指标采集压力过大时，可以创建新的 Prometheus 实例，将 scrape job 拆分给多个 Prometheus，Querier 从多个 Prometheus 查询汇聚结果，降低单个 Prometheus 的压力。

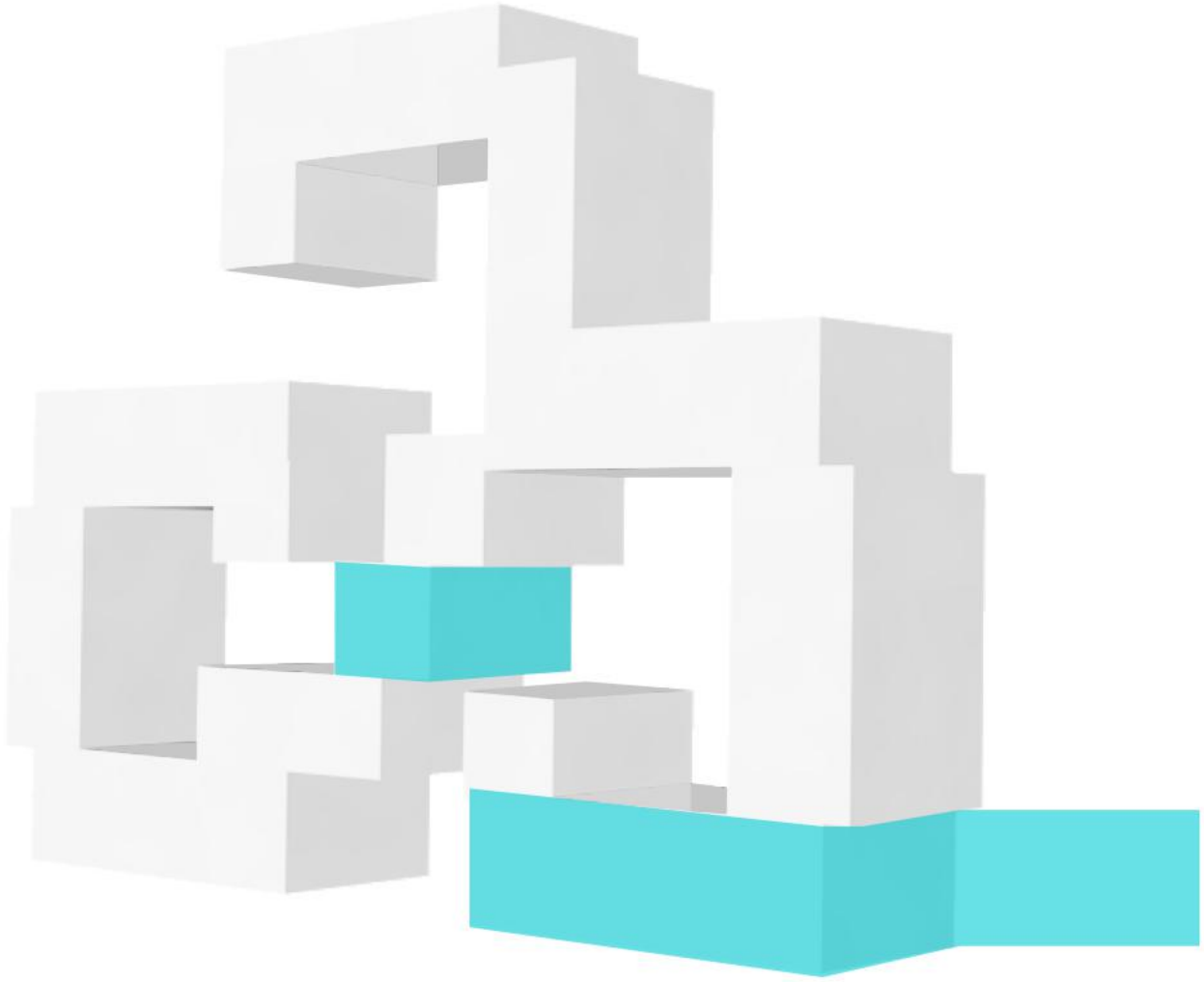
Q&A

备注：干货

长按添加小助手，获取资料

云智慧 Cloudwise | 全栈智能业务运维

感谢观看



4006661332
www.cloudwise.com