

轻松驾驭！Prometheus 如何监控指标，快速定位故障

🕒 2024-01-26 10:17 🏷️ 网络安全 👁️ 阅读 784 💬 评论 0



运维派隶属马哥教育旗下专业运维社区，是国内成立最早的IT运维技术社区，欢迎关注公众号：yunweipai
领取学习更多免费Linux云计算、Python、Docker、K8s教程关注公众号：马哥linux运维

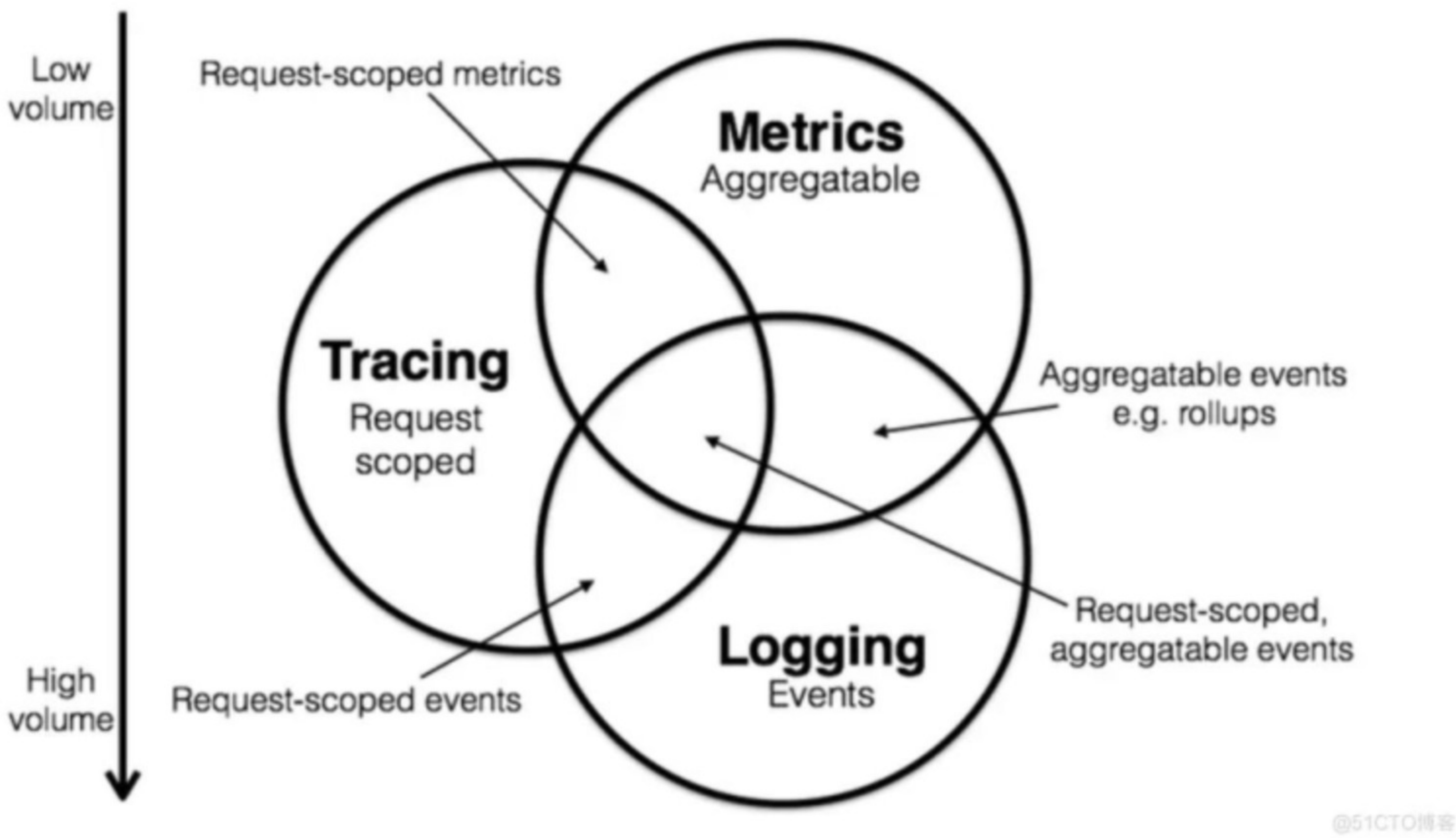
Prometheus 监控业务指标

在 Kubernetes 已经成了事实上的容器编排标准之下，微服务的部署变得非常容易。但随着微服务规模的扩大，服务治理带来的挑战也会越来越大。在这样的背景下出现了服务可观测性（observability）的概念。

在分布式系统里，系统的故障可能出现在任何节点，怎么能在出了故障的时候快速定位问题和解决问题，甚至是在故障出现之前就能感知到服务系统的异常，把故障扼杀在摇篮里。这就是可观测性的意义所在。

可观测性

可观测性是由 logging, metrics, tracing 构建的, 简称为可观测性三支柱。



- **Lgging**, 展现的是应用运行而产生的事件或者程序在执行的过程中间产生的一些日志，可以详细解释系统的运行状态，但是存储和查询需要消耗大量的资源。所以往往使用过滤器减少数据量。
- **Metrics**, 是一种聚合数值，存储空间很小，可以观察系统的状态和趋势，但对于问题定位缺乏细节展示。这个时候使用等高线指标等多维数据结构来增强对于细节的表现力。例如统计一个服务的 TBS 的正确率、成功率、流量等，这是常见的针对单个指标或者某一个数据库的。
- **Tracing**, 面向的是请求，可以轻松分析出请求中异常点，但与 logging 有相同的问题就是资源消耗较大。通常也需要通过采样的方式减少数据量。比如一次请求的范围，也就是从浏览器或者手机端发起的任何一次调用，一个流程化的东西，我们需要轨迹去追踪。

这篇文章讨论的主题就是可观测性中的 metrics。在 k8s 作为基础设施的背景下，我们知道 K8s 本身是个复杂的容器编排系统，它本身的稳定运行至关重要。与之相伴的指标监控系统 Promethues 已经成为了云原生服务下监控体系的事实标准。

相信大家对资源层面比如 CPU，Memory，Network；应用层面比如 Http 请求数，请求耗时等指标的监控都有所了解。那么业务层面的指标又怎么利用 Prometheus 去监控和告警呢？这就是这篇文章的核心内容。

以我们一个业务场景为例，在系统中有多种类型的 task 在运行，并且 task 的运行时间各异，task 本身有各种状态包括待执行、执行中、执行成功、执行失败等。如果想确保系统的稳定运行，我们必须对各个类型的 task 的运行状况了如指掌。比如当前是否有任务挤压，失败任务是否过多，并且当超过阈值是否告警。

为了解决上述的监控告警问题，我们先得了解一下 Prometheus 的指标类型

指标

指标定义

在形式上，所有的指标（Metric）都通过如下格式标示：

```
<metric name>{<label name>=<label value>, ...}
```

指标的名称（metric name）可以反映被监控样本的含义（比如，

```
http_request_total
```

— 表示当前系统接收到的HTTP请求总量）。指标名称只能由ASCII字符、数字、下划线以及冒号组成并必须符合正则表达式

```
[a-zA-Z_][a-zA-Z0-9_]*
```

。

标签（label）反映了当前样本的特征维度，通过这些维度Prometheus可以对样本数据进行过滤，聚合等。标签的名称只能由ASCII字符、数字以及下划线组成并满足正则表达式

```
[a-zA-Z_][a-zA-Z0-9_]*
```

。

指标类型

他的文章

发布文章
0

阅读数量

Linux工程师高薪必备大合集

90%的工程师偷偷在学



20本经典必备Linux+Python电子书



2022版学习路线图Linux+Python



近3年10家一线互联网面试真题



38张高薪IT工程师技能图谱



2022版Linux必会核心技能训练营



价值298元

名额有限 **立即领取 >>**



更有海量课程免费开放中
扫描左侧二维码
即可获得课程免费学习

马哥教育

热门文章

周 | 月 | 年

- 1 test10/02
- 2 Mostbet Tetbiqi Apk-ni Android-
- 3 100个shell高效命令全掌握，让
- 4 Linux中存储的基本管理



欢迎投稿

Prometheus定义了4种不同的指标类型（metric type）：Counter（计数器）、Gauge（仪表盘）、Histogram（直方图）、Summary（摘要）。

Counter

Counter类型的指标其工作方式和计数器一样，只增不减（除非系统发生重置）。常见的监控指标，如

```
http_requests_total
```

```
,
```

```
node_cpu
```

都是 Counter 类型的监控指标。一般在定义Counter类型指标的名称时推荐使用

```
_total
```

作为后缀。

通过 counter 指标我们可以和容易的了解某个事件产生的速率变化。

例如，通过

```
rate()
```

函数获取HTTP请求量的增长率：

```
rate(http_requests_total[5m])
```

Gauge

Gauge类型的指标侧重于反应系统的当前状态。因此这类指标的样本数据可增可减。常见指标如：

```
node_memory_MemFree
```

（主机当前空闲的内容大小）、

```
node_memory_MemAvailable
```

（可用内存大小）都是Gauge类型的监控指标。

通过Gauge指标，我们可以直接查看系统的当前状态

```
node_memory_MemFree
```

Summary

Summary 主用于统计和分析样本的分布情况。比如某 Http 请求的响应时间大多数都在 100 ms 内，而个别请求的响应时间需要 5s，那么这中情况下统计指标的平均值就不能反映出真实情况。而如果通过 Summary 指标我们能立马看响应时间的9分位数，这样的指标才是有意义的。

例如

```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 3.98e-05
go_gc_duration_seconds{quantile="0.25"} 5.31e-05
go_gc_duration_seconds{quantile="0.5"} 6.77e-05
go_gc_duration_seconds{quantile="0.75"} 0.0001428
go_gc_duration_seconds{quantile="1"} 0.0008099
go_gc_duration_seconds_sum 0.0114183
go_gc_duration_seconds_count 85
```

Histogram

Histogram 类型的指标同样用于统计和样本分析。与 Summary 类型的指标相似之处在于 Histogram 类型的样本同样会反应当前指标的记录的总数(以

```
_count
```

作为后缀)以及其值的总量（以

```
_sum
```

作为后缀）。不同在于 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len进行定义。同时对于Histogram的指标，可以通过

```
histogram_quantile()
```

函数计算出其值的分位数。

例如

```
# HELP prometheus_http_response_size_bytes Histogram of response size for HTTP requests.
# TYPE prometheus_http_response_size_bytes histogram
prometheus_http_response_size_bytes_bucket{handler="/",le="100"} 1
prometheus_http_response_size_bytes_bucket{handler="/",le="1000"} 1
prometheus_http_response_size_bytes_bucket{handler="/",le="10000"} 1
prometheus_http_response_size_bytes_bucket{handler="/",le="100000"} 1
prometheus_http_response_size_bytes_bucket{handler="/",le="1e+06"} 1
prometheus_http_response_size_bytes_bucket{handler="/",le="+Inf"} 1
prometheus_http_response_size_bytes_sum{handler="/"} 29
prometheus_http_response_size_bytes_count{handler="/"} 1
```

应用指标监控

暴露指标

Prometheus 最常用的方式是通过 pull 去抓取 metrics。所以我们首先在服务通过

```
/metrics
```

接口暴露指标，这样 Promethues server 就能通过 http 请求抓取到我们的业务指标。

接口示例


```
server := gin.New()
server.Use(middlewares.AccessLogger(), middlewares.Metric(), gin.Recovery())

server.GET("/health", func(c *gin.Context) {
    c.JSON(http.StatusOK, gin.H{
        "message": "ok",
    })
})

server.GET("/metrics", Monitor)func Monitor(c *gin.Context) {
    h := promhttp.Handler()
    h.ServeHTTP(c.Writer, c.Request)
}
```

定义指标

为了方便理解，这里选取了三种类型和两种业务场景的指标
示例

```
var (
    //HTTPReqDuration metric:http_request_duration_seconds
    HTTPReqDuration *prometheus.HistogramVec
    //HTTPReqTotal metric:http_request_total
    HTTPReqTotal *prometheus.CounterVec
    // TaskRunning metric:task_running
    TaskRunning *prometheus.GaugeVec
)

func init() {
    // 监控接口请求耗时
    // 指标类型是 Histogram
    HTTPReqDuration = prometheus.NewHistogramVec(prometheus.HistogramOpts{
        Name:  "http_request_duration_seconds",
        Help:  "http request latencies in seconds",
        Buckets: nil,
    }, []string{"method", "path"})
    // "method"、"path" 是 label

    // 监控接口请求次数
    // 指标类型是 Counter
    HTTPReqTotal = prometheus.NewCounterVec(prometheus.CounterOpts{
        Name: "http_requests_total",
        Help: "total number of http requests",
    }, []string{"method", "path", "status"}) // "method"、"path"、"status" 是 label

    // 监控当前正在执行的 task 数量
    // 监控类型是 Gauge
    TaskRunning = prometheus.NewGaugeVec(prometheus.GaugeOpts{
        Name: "task_running",
        Help: "current count of running task",
    }, []string{"type", "state"})
    // "type"、"state" 是 label

    prometheus.MustRegister(
        HTTPReqDuration,
        HTTPReqTotal,
        TaskRunning,
    )
}
```

通过上述的代码我们就定义并且注册了我们的想要监控的指标。

生成指标

示例

```
start := time.Now()
c.Next()

duration := float64(time.Since(start)) / float64(time.Second)

path := c.Request.URL.Path

// 请求数加1
controllers.HTTPReqTotal.With(prometheus.Labels{
    "method": c.Request.Method,
    "path":   path,
    "status": strconv.Itoa(c.Writer.Status()),
}).Inc()

// 记录本次请求处理时间
controllers.HTTPReqDuration.With(prometheus.Labels{
    "method": c.Request.Method,
    "path":   path,
}).Observe(duration)

// 模拟新建任务
controllers.TaskRunning.With(prometheus.Labels{
    "type": shuffle([]string{"video", "audio"}),
    "state": shuffle([]string{"process", "queue"}),
}).Inc()

// 模拟任务完成
controllers.TaskRunning.With(prometheus.Labels{
    "type": shuffle([]string{"video", "audio"}),
    "state": shuffle([]string{"process", "queue"}),
}).Dec()
```

抓取指标

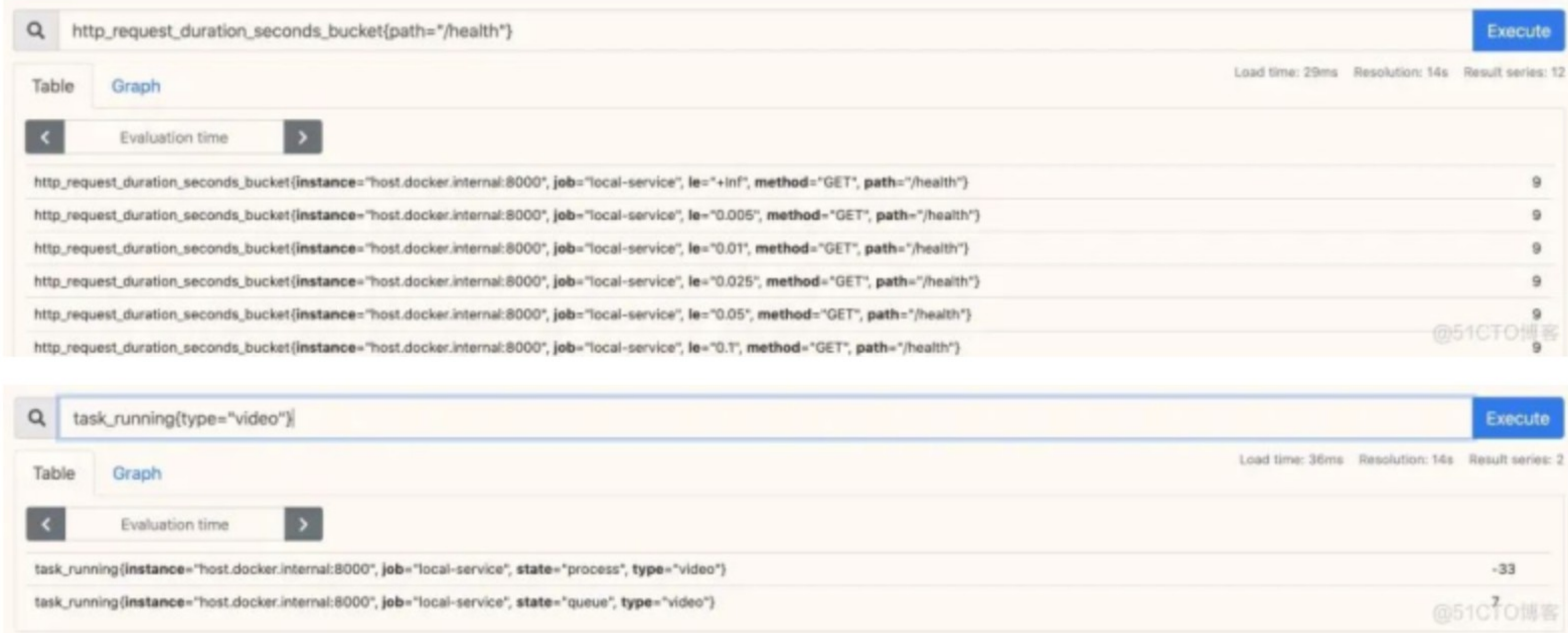
Promethues 抓取 target 配置


```
# 抓取间隔
scrape_interval: 5s


# 目标
scrape_configs:
- job_name: 'prometheus'
  static_configs:
    - targets: ['prometheus:9090']
- job_name: 'local-service'
  metrics_path: /metrics
  static_configs:
    - targets: ['host.docker.internal:8000']
```

在实际应用中静态配置 target 地址不太适用，在 k8s 下 Promethues通过与 Kubernetes API 集成目前主要支持5种服务发现模式，分别是：Node、Service、Pod、Endpoints、Ingress。

指标展示如下图：



链接：https://blog.51cto.com/u_15576159/8017962

（版权归原作者所有， 侵删）

本文链接：<https://www.yunweipai.com/44575.html>

点赞 0

分享到微信 分享到微博

上一篇：信息安全工程师需要掌握哪些能力？

下一篇：华为大佬分享的《Linux命令大全》太赞了！（完整版PDF限时下载）



相关文章 | RELATED POSTS

- 1月8号-9号【网安实训营】，手把手教你拿下某微商城

Mao 网络安全 阅读 284 2025-01-08 15:11
- 紧急提醒：病毒变种，多个微信群已出现！千万不要点开！

doubao 网络安全 阅读 436 2024-11-29 17:33
- 零基础学网安，入门必看的5本书籍（附PDF）

Mao 网络安全 阅读 855 2024-10-14 15:10
- 渗透测试报告一键生成工具

Gavin 网安资源 阅读 696 2024-09-29 14:27
- 3天光速成为一名黑客，看这个教程就够了！

doubao 网安资源 阅读 604 2024-09-10 14:08
- 9月4日20:00，免费公开课【60分钟了解勒索病毒】已有875人报名，速来

doubao 网安资源 阅读 604 2024-09-04 10:27
- 2024参加护网-必备知识宝典（附完整版PDF）

Gavin 网络安全 阅读 895 2024-08-08 17:20
- 国家入编新职业：网络安全等级保护测评师！


Mao 网安资源 阅读 1.3k 2024-08-08 17:08
- 牛！0.1元学会网络安全Oday漏洞攻击猎捕！还能领实体书！


Yong Yue 网安资源 阅读 634 2024-07-08 10:27
- 阿里悬赏 500万，需要这类人才！


Mao 网安资源 阅读 634 2024-07-08 10:18

网友评论 | COMMENTS

发表回复

 名称（必填）

 邮箱（必填）

 网址（选填）

发表评论

暂无评论