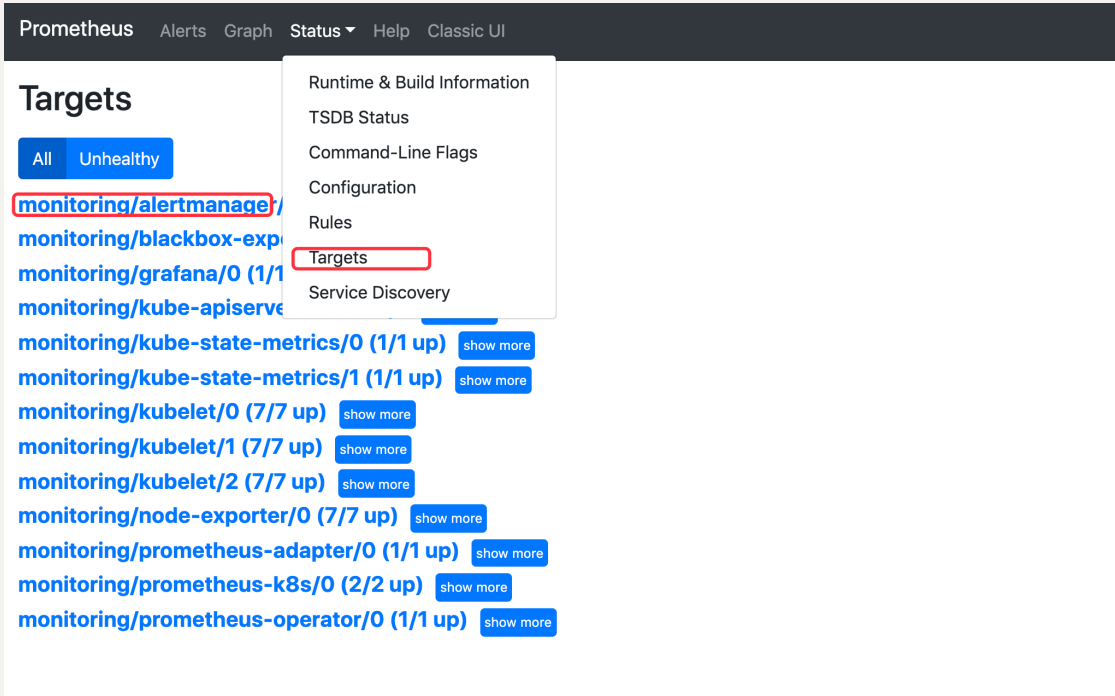


这里我就会告诉大家一些常用的Prometheus语法函数

数据信息都是从metrics接口中获取到的



monitoring/alertmanager/0 (3/3 up) [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.42.0.4:9093/metrics	UP	<code>container="alertmanager"</code> <code>endpoint="web"</code> <code>instance="10.42.0.4:9093"</code> <code>job="alertmanager-main"</code> <code>namespace="monitoring"</code> <code>pod="alertmanager-main-0"</code> <code>service="alertmanager-main"</code>	50.904s	3.059ms	
http://10.42.5.5:9093/metrics	UP	<code>container="alertmanager"</code> <code>endpoint="web"</code> <code>instance="10.42.5.5:9093"</code> <code>job="alertmanager-main"</code> <code>namespace="monitoring"</code> <code>pod="alertmanager-main-1"</code> <code>service="alertmanager-main"</code>	1m 3s	3.521ms	
http://10.42.6.9:9093/metrics	UP	<code>container="alertmanager"</code> <code>endpoint="web"</code> <code>instance="10.42.6.9:9093"</code> <code>job="alertmanager-main"</code> <code>namespace="monitoring"</code> <code>pod="alertmanager-main-2"</code> <code>service="alertmanager-main"</code>	47.801s	3.356ms	

```
# 可以在容器执行curl获取到这些数据
curl http://10.42.0.4:9093/metrics
```

我们现在把官方这些rule配置全部删除掉，我们自己去写报警规则,我们需怎么怎么删除呢，Prometheus-Operator中的报警规则，不是configmap 或者secret 之类的来进行加载的，而是通过CRD资源类PrometheusRule来生成的，所以如果以后我们需要自定义一个报警选项的话，只需要定义一个 PrometheusRule 资源对象即可。

但是为什么 Prometheus 能够识别这个 PrometheusRule 资源对象呢？这就需要查看我们创建的 prometheus 这个资源对象了，里面有非常重要的一个属性 **ruleSelector**，用来匹配 rule 规则的过滤器，要求匹配具有 prometheus=k8s 和 role=alert-rules 标签的 PrometheusRule 资源对象，

然后呢我们创建一个 PrometheusRule 资源对象后，会自动在上面的 prometheus-k8s-rulefiles-0 目录下面生成一个对应的 <namespace>-<name>.yaml 文件

```
apiVersion: monitoring.coreos.com/v1
kind: Prometheus
metadata:
  labels:
    prometheus: k8s
  name: k8s
  namespace: monitoring
spec:
  alerting:
    alertmanagers:
      - name: alertmanager-main
        namespace: monitoring
        port: web
  image: quay.io/prometheus/prometheus:v2.15.2
  nodeSelector:
    kubernetes.io/os: linux
  podMonitorNamespaceSelector: {}
  podMonitorSelector: {}
  replicas: 2
  retention: 3h
  podMetadata:
    labels:
      thanos-store-api: "true"
  resources:
    requests:
      cpu: 500m
      memory: 10Gi
    limits:
      cpu: 3500m
      memory: 15Gi
  ruleSelector:
    matchLabels:
      prometheus: k8s
      role: alert-rules
```

#我们现在来删除官方规则

```
find . -name '*Rule*' | xargs -n 1 kubectl delete -f
```

自己写规则

Prometheus的查询语言比较丰富包含着，支持条件查询、算术运算、比较操作符、正则表达式、聚合操作符、逻辑操作符。

类型	操作符	示例
比较操作符	<ul style="list-style-type: none">• = 等于• != 不等于• > 大于• < 小于• >= 大于等于• <= 小于等于	<code>node_cpu_seconds_total{job="Linux Server",mode="iowait"}</code> <code>node_cpu_seconds_total{job="Linux Server",mode=~"user system"}</code> <code>node_cpu_seconds_total{job="Linux Server",mode=~"user system",cpu!="0"}</code>
算术操作符	<ul style="list-style-type: none">• + 加法• - 减法• * 乘法• / 除法	CPU使用率: <code>100 - (avg(irate(node_cpu_seconds_total{mode="idle"}[5m])) by (instance) * 100)</code> 内存使用率: <code>100 - (node_memory_MemFree_bytes+node_memory_Cached_bytes+node_memory_Buffers_bytes) / node_memory_MemTotal_bytes * 100</code>
正则匹配操作符	<ul style="list-style-type: none">• =~ 正则表达式匹配• !~ 正则表达式匹配结果取反	磁盘使用率: <code>100 - (node_filesystem_free_bytes{mountpoint="/",fstype=~"ext4 xfs"} / node_filesystem_size_bytes{mountpoint="/",fstype=~"ext4 xfs"} * 100)</code>
聚合操作符	<ul style="list-style-type: none">• sum (在维度上求和)• max (在维度上求最大值)• min (在维度上求最小值)• avg (在维度上求平均值)• count(统计样本数量)	所有实例CPU system使用率总和: <code>sum(node_cpu_seconds_total{job="Linux Server",mode="system"})</code> 所有实例CPU system变化速率平均值: <code>avg(irate(node_cpu_seconds_total{job="Linux Server",mode="system"}[5m]))</code> 统计CPU数量: <code>count(node_cpu_seconds_total{job="Linux Server",mode="system"})</code>
逻辑操作符	<ul style="list-style-type: none">• and 与• or 或	大于10并且小于50: <code>prometheus_http_requests_total > 10 and prometheus_http_requests_total < 50</code> 大于10或者小于50: <code>prometheus_http_requests_total > 10 or prometheus_http_requests_total < 50</code>

首先我么来看下基本的语句写法

```
以为container_memory_rss例子：
#正常搜索显示全部数据
container_memory_rss

#在大扩号中放入字段中的语句可以单独查询，多个由逗号分隔
container_memory_rss{pod="rancher-operator-f54cf4887-c9px5",node="k8s-node-06"}

#比较操作符（常用于报警）
container_memory_rss{pod="rancher-operator-f54cf4887-c9px5",node="k8s-node-06"} < 50000

#算数操作符（这个语句就是实验，实际+1没什么功效，一般是将查询的多个数据进行一个算数聚合用，大家知道逻辑写法就行了）后面我都会实际用到
container_memory_rss{pod="rancher-operator-f54cf4887-c9px5",node="k8s-node-06"} +1
container_memory_rss{pod="rancher-operator-f54cf4887-c9px5",node="k8s-node-06"} /1024/1024

#正则匹配
#模糊匹配
```

```

    container_memory_rss{pod=~"alertmanager.*"} # =~需要配合.*使用,
否则数据出不来
    container_memory_rss{pod=~"alertmanager.*",node=~".*07"}
#不等于模糊匹配
    container_memory_rss{pod!~"alertmanager.*"}
#需要记住.* 必须要和~或者!~配合使用, 和= 或者 != 是不无法配合的如下面的语
句无法过滤alertmanager
    container_memory_rss{pod!="alertmanager.*"}

#聚合操作符求和
sum(container_memory_rss)
#按照pod为分组查询出每个pods当前使用的内存值
sum(container_memory_rss) by (pod)

#cpu数量统计
count(node_cpu_seconds_total) by (pod)

#平均增长量的话我们下面函数会用到

#逻辑操作符
#这个是and (两行写在一起)
container_memory_rss{pod="rancher-operator-f54cf4887-c9px5"} >
50000 and container_memory_rss{pod="rancher-operator-f54cf4887-
c9px5"} > 32276480

#这个是or (两行写在一起)
container_memory_rss{pod="rancher-operator-f54cf4887-c9px5"} >
50000 or container_memory_rss{pod="rancher-operator-f54cf4887-
c9px5"} > 32276480

```

常用函数

讲到函数之前我们需要了解下prometheus收集metrics的数据类型

Prometheus Metrics 有四种基本的 type:

#Counter:计数器, 用于统计类似于: CPU时间, API访问总次数, 异常发生次数等等场景。这些指标的特点就是增加不减少。

- Counter: 只增不减的单变量

#Gauge:计量器, 适用于如: 当前内存使用率, 当前CPU使用率, 特点是数值是可以增加或者减少, 计量器是监控中大部分数据类型

- Gauge: 可增可减的单变量

#Histogram柱状图: 用于统计一些数据分布的情况

- Histogram: 多桶统计的多变量

#Summary: 与柱状图类似, 主要用于计算在一定时间窗口范围内度量指标对象的总数以及所有对量指标值的总和。

- Summary: 聚合统计的多变量

通过curl <http://10.42.5.5:9093/metrics>可以看见数据类型, 如图:

≥ 命令行: rancher

已连接

高级技巧: 点击运行命令行时按住 Command 键在新窗口中打开

```
csot@rancher-6666c8454-bfzwl:/var/lib/rancher# curl http://10.42.5.5:9093/metric
# HELP alertmanager_alerts How many alerts by state.
# TYPE alertmanager_alerts gauge
alertmanager_alerts{state="active"} 0
alertmanager_alerts{state="suppressed"} 0
# HELP alertmanager_alerts_invalid_total The total number of received alerts that were invalid.
# TYPE alertmanager_alerts_invalid_total counter
alertmanager_alerts_invalid_total{version="v1"} 0
alertmanager_alerts_invalid_total{version="v2"} 0
# HELP alertmanager_alerts_received_total The total number of received alerts.
# TYPE alertmanager_alerts_received_total counter
alertmanager_alerts_received_total{status="firing",version="v1"} 0
alertmanager_alerts_received_total{status="firing",version="v2"} 11415
alertmanager_alerts_received_total{status="resolved",version="v1"} 0
alertmanager_alerts_received_total{status="resolved",version="v2"} 94
# HELP alertmanager_build_info A metric with a constant '1' value labeled by version, revision, branch, and goversion from which ale
rtmanager was built.
# TYPE alertmanager_build_info gauge
alertmanager_build_info{branch="HEAD",goversion="go1.14.4",revision="4c6c03ebfe21009c546e4d1e9b92c371d67c021d",version="0.21.0"} 1
# HELP alertmanager_cluster_alive_messages_total Total number of received alive messages.
# TYPE alertmanager_cluster_alive_messages_total counter
alertmanager_cluster_alive_messages_total{peer="01EZV3K93VNV1H0S7NQE3YQFC"} 14046
alertmanager_cluster_alive_messages_total{peer="01EZV3KBFQ9VMG4W272K8KE"} 14046
alertmanager_cluster_alive_messages_total{peer="01EZV3KBPXWKG1470NGG274"} 14045
# HELP alertmanager_cluster_enabled Indicates whether the clustering is enabled or not.
# TYPE alertmanager_cluster_enabled gauge
alertmanager_cluster_enabled 1
# HELP alertmanager_cluster_failed_peers Number indicating the current number of failed peers in the cluster.
# TYPE alertmanager_cluster_failed_peers gauge
alertmanager_cluster_failed_peers 0
# HELP alertmanager_cluster_health_score Health score of the cluster. Lower values are better and zero means 'totally healthy'.
# TYPE alertmanager_cluster_health_score gauge
alertmanager_cluster_health_score 0
# HELP alertmanager_cluster_members Number indicating current number of members in cluster.
# TYPE alertmanager_cluster_members gauge
alertmanager_cluster_members 3
# HELP alertmanager_cluster_messages_pruned_total Total number of cluster messages pruned.
# TYPE alertmanager_cluster_messages_pruned_total counter
alertmanager_cluster_messages_pruned_total 0
# HELP alertmanager_cluster_messages_queued Number of cluster messages which are queued.
# TYPE alertmanager_cluster_messages_queued gauge
alertmanager_cluster_messages_queued 0
# HELP alertmanager_cluster_messages_received_size_total Total size of cluster messages received.
# TYPE alertmanager_cluster_messages_received_size_total counter
alertmanager_cluster_messages_received_size_total{msg_type="full_state"} 196560
alertmanager_cluster_messages_received_size_total{msg_type="update"} 0
```

Prometheus 函数

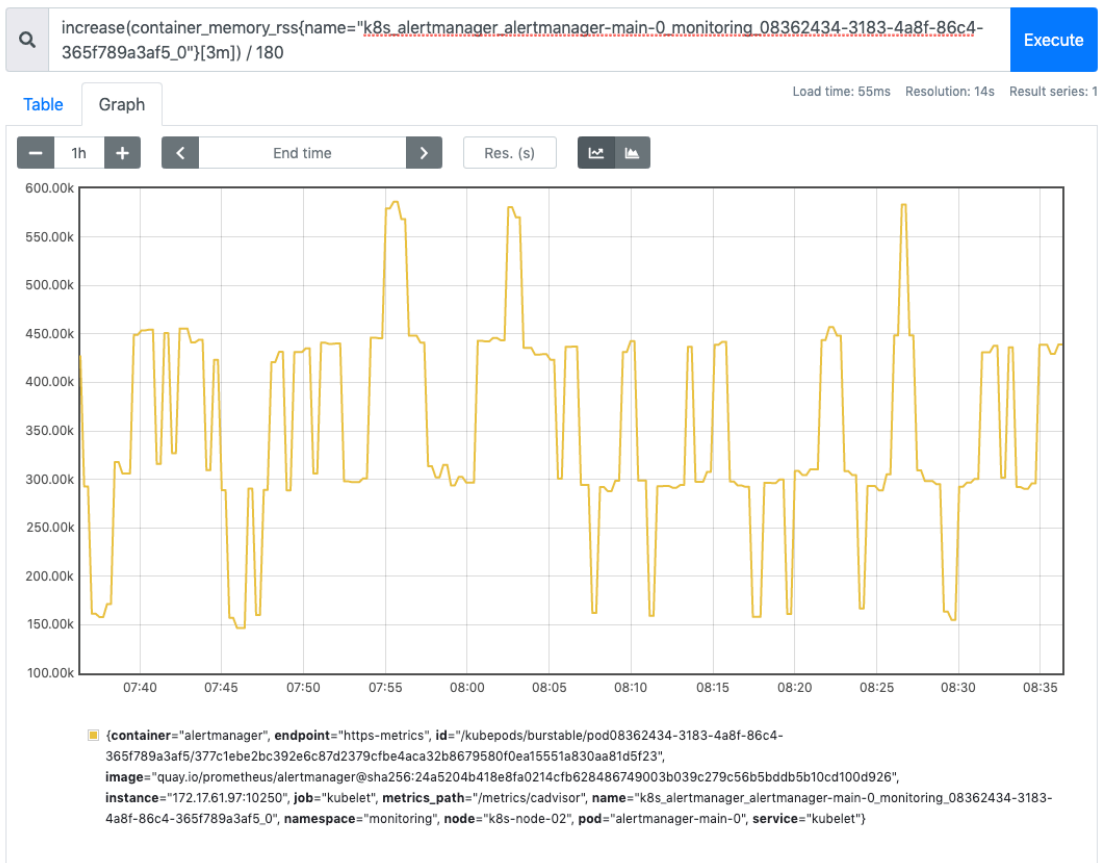
#increase函数：获取区间向量中的第一个和最后一个样本并返回其增长量。如果除以一定时间就可以获取该时间内的平均增长率。该函数需配合计数器类型数据（counter）一起使用，遇到counter数据类型，在做任何操作之前先套rate()或者increase()函数。

```
increase(container_memory_rss{name="k8s_alertmanager_alertmanager-  
-main-0_monitoring_08362434-3183-4a8f-86c4-365f789a3af5_0"}[2m])  
/ 120
```

#rate()函数：获取指定时间段的平均量。比如container2分钟期间累计量增加了1000k，加入rate([2m])函数后就会使用1000除以60秒，计算出数据大约为16bytes。该函数需配合计数器类型数据（counter）一起使用，遇到counter数据类型，在做任何操作之前先套rate()或者increase()函数。

```
rate(container_memory_rss{name="k8s_alertmanager_alertmanager-  
-main-0_monitoring_08362434-3183-4a8f-86c4-365f789a3af5_0"}[2m])
```

上面两个出的数据是一样的如下图：得出3分钟平均增涨率为150-600k，但是increase需要做个除操作：如[2m]需要除以120 [5m]除以300



[Remove Panel](#)



topk函数：该函数可取出排行前几的数据，N可以自定义。比如监控内存使用率
container_memory_rss可以取出前几个排名
topk(10,container_memory_rss)

#label_replace函数：对label进行二次处理，生成一个我们自定义的label，删除或者增加不想要的字符串
#以prometheus_http_requests_total为例子字段中会有
instance="10.42.2.11:9090" label字段，我感觉这个字段英文的意思不太贴切，我想给他生成一个我自定义的label字段叫metricsIP，我需要按照以下来做。就能生成一个metricsIP自定义字段#"metricsIP"代表着新字段名称，\$1代表新字段数据内容
取instance等于的所有数据(.*)

```
label_replace(prometheus_http_requests_total, "metricsIP", "$1",  
,"instance" , "(.*)")
```

#如果想要去除instance="10.42.2.11:9090"中的9090端口,使用以下代表以:为分隔符，取第一个字符串10.42.2.11

```
label_replace(prometheus_http_requests_total, "metricsIP", "$1",  
,"instance" , "(.*):(.*)")
```

#结合上面可以更改任何数据格式如：变成了metricsIP="10.42.2.11-9090"

```
label_replace(prometheus_http_requests_total, "metricsIP", "$1-$2",  
,"instance" , "(.*):(.*)")
```

label_replace函数适用于更改数据格式，如果有的数据特别长，我们需要截取一段就可以使用这个方法

案例2：

#node_load5代表5分钟CPU负载，machine_cpu_cores代表CPU核心数，下面语句算出了当前CPU使用率，因为node_load5中主机label为instance，而machine_cpu_cores为node，不同label不能相除，所以改node_load5的instance为node就能相除了。

```
sum(label_replace(node_load5, "node", "$1", "instance", "(.*)"))  
by (node)/ sum(machine_cpu_cores) by (node)*100 >30
```



```
#predict_linear函数:对曲线变化速率进行计算,起到一定的预测作用。比如当前这6
个小时的磁盘可用率急剧下降,这种情况可能导致磁盘很快被写满,这时可以使用该函
数,用当前6小时的数据增涨速率去预测未来24个小时的状态,实现提前告警以
node_filesystem_avail_bytes (数据返回可用磁盘空间大小) 为例子:
predict_linear(node_filesystem_avail_bytes{job="node-
exporter",fstype!=""}[6h], 24*60*60)
```

```
#abs 函数:返回输入向量的样本的绝对值,这个是什么意思呢,这个一般是判断指定程序
是否存活的,比如有个数据是up
#sql语句直接查询up, up的metric能够标识目标的状态。如果值是1,那么能够成功
的从目标抓取metrics; 如果值是0,那么则标明从该目标抓取metrics失败。
up
#然后使用abs获取up中metric状态,如果等于0就证明抓取失败就会进行报警
abs(up{job="kubelet", metrics_path="/metrics"} == 0)
```

#absent函数: 如果赋值给它的向量具有样本数据, 则返回空向量; 如果传递的瞬时向量参数没有样本数据, 则返回不带度量指标名称且带有标签的样本值为1的结果, 当监控度量指标时, 如果获取到的样本数据是空的, 使用absent方法对告警是非常有用的, 下面例子就没有返回, 因为absent设定的值为数据的正确值1, 如果metrics数据值为1就不会进行报警, 和abs是相反的。

```
absent(up{job="kubelet", metrics_path="/metrics"} == 1)
```

#而且如果我新的集群里面想加一个kafka的监控, 我忘记了, 但是我这条规则在就相当于能提醒我你忘记加kafka监控了, 下面的值会返回1, 就会报警。

```
absent(up{job="kafka", metrics_path="/metrics"} == 1) 1
```

常用函数就到这里吧, Prometheus函数有很多, 活学活用就可以了, 然后我们刚才删除了所有的rule规则, 我们来自自己写几个规则搞一搞。

Pods内存计算规则

container_memory_rss	gauge	字节数 bytes	RSS内存，即常驻内存集（Resident Set Size），是分配给进程使用实际物理内存，而不是磁盘上缓存的虚拟内存。RSS内存包括所有分配的栈内存和堆内存，以及加载到物理内存中的共享库占用的内存空间，但不包括进入交换分区的内存。
container_memory_usage_bytes	gauge	字节数 bytes	当前使用的内存量，包括所有使用的内存，不管有没有被访问。
container_memory_max_usage_bytes	gauge	字节数 bytes	最大内存使用量的记录。
container_memory_cache	gauge	字节数 bytes	高速缓存（cache）的使用量。cache是位于CPU与主内存间的一种容量较小但速度很高的存储器，是为了提高cpu和内存之间的数据交换速度而设计的。
container_memory_swap	gauge	字节数 bytes	虚拟内存使用量。虚拟内存（swap）指的是用磁盘来模拟内存使用。当物理内存快要使用完或者达到一定比例，就可以把部分不用的内存数据交换到硬盘保存，需要使用时再调入物理内存
container_memory_working_set_bytes	gauge	字节数 bytes	当前内存工作集（working set）使用量。
container_memory_failcnt	counter	次	申请内存失败次数计数
container_memory_failures_total	counter	次	累计的内存申请错误次数

#container cpu使用率

```
max by(pod) (rate(container_cpu_usage_seconds_total[3m])) / max
by(pod) (container_spec_cpu_quota / 100000) * 100
```

```
# container内存使用率
#取最大设置资源的pod ,如果一个pod中有多个容器,pods中是k8s中最小的调度单元,
一个pod中可能会有很多容器, metric数据中container_memory_rss统计的是所有容器内存值的聚合
max by(pod)(container_memory_rss) / max by(pod)
(container_spec_memory_limit_bytes) * 100 != +inf

# 或者是直接获取
100 * (sum by(pod, namespace)
(container_memory_working_set_bytes) / 2 /
      sum by(pod, namespace)
(label_replace(kube_pod_container_resource_limits_memory_bytes{pod!="alertmanager-main-0"}, "pod_name", "$1", "pod", "(.*)")))
```

```
# pod重启次数
sum (increase (kube_pod_container_status_restarts_total{}[1m]))
by (namespace,pod) >0
```

```
# 检查非正常状态的pod
sum(kube_pod_status_phase{phase!~"Running|Succeeded"}) by
(phase,pod,namespace)
```

```
#根文件系统使用率,需要100 - 因为计算出来的是磁盘剩余量
100 - ((node_filesystem_avail_bytes{mountpoint="/",
fstype!="ext4"} * 100) / node_filesystem_size_bytes
{mountpoint="/"})
```

```
#有状态服务数量少于百分之70
100 * (sum (kube_statefulset_status_replicas_ready) by
(statefulset) / sum (kube_statefulset_status_replicas) by
(statefulset)) < 70
```

语句咱们就说到这里了，同学们可以找到<https://github.com/kubernetes/kube-state-metrics/tree/master/docs> 下所有的参数及属性来写自己需求的SQL

PrometheusRule配置

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  labels:
    prometheus: k8s
    role: alert-rules
  name: pods-rules
  namespace: monitoring
spec:
  groups:
    - name: pods.rules
      rules:
        - alert: PodMemory>90%
          annotations:
            description: 当前使用率 {{ printf "%.2f" $value }}%
            summary: 容器cpu使用率高于百分之90
          expr: |
            max by(pod) (rate(container_cpu_usage_seconds_total[3m]))
            / max by(pod) (container_spec_cpu_quota / 100000) * 100 > 90
          for: 1m
          labels:
            severity: critical
```

邮件告警配置：找到Alertmanager的Secret

```
apiVersion: v1
kind: Secret
metadata:
  labels:
    alertmanager: main
    app.kubernetes.io/component: alert-router
    app.kubernetes.io/name: alertmanager
    app.kubernetes.io/part-of: kube-prometheus
    app.kubernetes.io/version: 0.21.0
  name: alertmanager-main
  namespace: monitoring
stringData:
```

```

alertmanager.yaml: |-
  global:
    resolve_timeout: 1m
    smtp_smarthost: 'smtp.126.com:465'
    smtp_from: 'kaikebak8s@126.com'
    smtp_auth_username: 'kaikebak8s@126.com'
    smtp_auth_password: 'UTXGEJFQTCJNDAHQ'
    smtp_require_tls: false
  route:
    group_by: ['alertname']
    group_wait: 10s # 当一个新的告警触发时, 这个告警要等待group_wait的
    时间直到调度通知
    group_interval: 10s # 组规则触发后, 要等10秒才能发出去,如果触发了
    相同一个组内的alert打包一起发
    repeat_interval: 1m # 当上次发送通知到现在的间隔大于
    repeat_interval
    receiver: 'mail'
    receivers:
    - name: 'mail'
      email_configs:
      - to: 'kaikebak8s@126.com,377239936@qq.com'
        send_resolved: true
    #报警一般分为级别规则严重 (critical), 警告 (warning), 正常(info),
    如果通一个alertname规则同时出发了严重和警告他会只报严重规则的, 而不报警告规
    则, 例子就是: 假如我规则配置了磁盘报警超过了百分之90, 和百分之80, 超过百分之
    90的是严重, 超过百分之80的警告, 如果磁盘瞬间到达了百分之90, 就只报报警超过了
    百分之90的规则了, 因为如果同时报出来的话没有意义, 下面字段为配置
    inhibit_rules:
    - source_match: #当此级别告警发生, 其他的告警被抑制
        severity: 'critical'
        target_match: # 被抑制的对象
        severity: 'warning'
        equal: ['alertname','target','job','instance']
  type: Opaque

```

钉钉告警配置:

```

###部署02webhook-secret.yaml #这里下面配置2个webhook (webhook和
base) 方便配置分群业务报警

```

```

apiVersion: v1
data: {}
kind: Secret
metadata:
  name: webhook-config
  namespace: monitoring
type: Opaque
stringData:
  config.yml: |-
    targets:
      webhook:
        url: "https://oapi.dingtalk.com/robot/send?
access_token=581d337655eb5ef1407e65acf83e07b6f2e8f05b0f2007e25f70
7028d147ab73"
        secret:
SECa08ddf4c65f241c46c2c8f3b7efb902cc0d7783ef1c18d3d72b8daleda9310
34
        message:
          title: '{{ template "ding.link.title" . }}'
          text: '{{ template "ding.link.content" . }}'
        mention:
          all: true
      base:
        url: "https://oapi.dingtalk.com/robot/send?
access_token=644e2ac7a337b954c2a344dff3948e044da156c97ab91a80560e
6d5b6eb78d08"
        secret:
SEC8c078b09172dfa29d670a876c35d55881e050bc73ef5897d6d1fd853a114db
90
        message:
          title: '{{ template "ding.link.title" . }}'
          text: '{{ template "ding.link.content" . }}'
        mention:
          all: true
      templates:
      - /etc/prometheus-webhook-dingtalk/templates/default.tpl
---
apiVersion: v1
data: {}
kind: Secret
metadata:
  name: webhook-template
  namespace: monitoring
type: Opaque
stringData:
  default.tpl: |-

```

```

    {{ define "__subject" }}{{ .Status | toUpper }}{{ if eq
.Status "firing" }}:{{ .Alerts.Firing | len }}{{ end }} {{
.GroupLabels.SortedPairs.Values | join " " }} {{ if gt (len
.CommonLabels) (len .GroupLabels) }} ({{ with .CommonLabels.Remove
.GroupLabels.Names }}{{ .Values | join " " }}{{ end }}){{ end }}
{{ end }}

    {{ define "__alertmanagerURL" }}{{
"http://59.110.71.101:59999/alerts" }}/#/alerts?receiver={{
.Receiver }}{{ end }}

    {{ define "default.__text_alert_list" }}{{ range . }}
    ---
    **集群系统:** <font color="#dd0000">KKB-PROD</font><br/>

    **开课吧负责团队:** <font color="#dd0000">猫头鹰</font><br/>

    **告警级别:** <font color="#dd0000">{{ .Labels.severity | upper
}}</font><br/>

    **触发时间:** <font color="#dd0000">{{ dateInZone "2006.01.02
15:04:05" (.StartsAt) "Asia/Shanghai" }}</font><br/>

    **当前状态:** <font color="#dd0000">未恢复 [打叉]</font><br/>

    **事件信息:**

    {{ range .Annotations.SortedPairs }} - {{ .Name }}: <font
color="#006666">{{ .Value | markdown | html }}</font><br/>
    {{ end }}

    **事件标签:**

    {{ range .Labels.SortedPairs }}{{ if and (ne (.Name)
"severity") (ne (.Name) "summary") (ne (.Name) "team") }} - {{
.Name }}: <font color="#006666">{{ .Value | markdown | html }}
</font><br/>
    {{ end }}{{ end }}
    {{ end }}
    {{ end }}

    {{ define "default.__text_alertresovle_list" }}{{ range . }}
    ---
    **集群系统:** <font color="#dd0000">KKB-PROD</font><br/>

    **开课吧负责团队:** <font color="#dd0000">开课吧运维组</font><br/>


```

```

    **告警级别:** <font color="#dd0000">{{ .Labels.severity | upper
}}</font><br/>

    **触发时间:** <font color="#dd0000">{{ dateInZone "2006.01.02
15:04:05" (.StartsAt) "Asia/Shanghai" }}</font><br/>

    **结束时间:** <font color="#00dd00">{{ dateInZone "2006.01.02
15:04:05" (.EndsAt) "Asia/Shanghai" }}</font><br/>

    **当前状态:** <font color="#dd0000">已恢复 </font><br/>

    **事件信息:**
    {{ range .Annotations.SortedPairs }} - {{ .Name }}: <font
color="#006666">{{ .Value | markdown | html }}</font><br/>
    {{ end }}

    **事件标签:**
    {{ range .Labels.SortedPairs }}{{ if and (ne (.Name)
"severity") (ne (.Name) "summary") (ne (.Name) "team") }} - {{
.Name }}: <font color="#006666">{{ .Value | markdown | html }}
</font><br/>
    {{ end }}{{ end }}
    {{ end }}
    {{ end }}

    {{/* Default */}}
    {{ define "default.title" }}{{ template "__subject" . }}{{
end }}
    {{ define "default.content" }}#### \[{{ .Status | toUpper }}
{{ if eq .Status "firing" }}:{{ .Alerts.Firing | len }}{{ end
}}\] **[{{ index .GroupLabels "alertname" }}]({{ template
"__alertmanagerURL" . }})**
    {{ if gt (len .Alerts.Firing) 0 -}}

    {{ template "default.__text_alert_list" .Alerts.Firing }}
    {{- end }}

    {{ if gt (len .Alerts.Resolved) 0 -}}
    {{ template "default.__text_alertresovle_list"
.Alert.Resolved }}
    {{- end }}
    {{- end }}

```



```
    {{/* Following names for compatibility */}}
    {{ define "ding.link.title" }}{{ template "default.title" .
}}{{ end }}
    {{ define "ding.link.content" }}{{ template "default.content"
. }}{{ end }}
```

```
#部署03webhook-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webhook-dingtalk
  namespace: monitoring
spec:
  selector:
    matchLabels:
      app: webhook-dingtalk
  template:
    metadata:
      labels:
        app: webhook-dingtalk
    spec:
      containers:
        - name: webhook-dingtalk
          image: timonwong/prometheus-webhook-dingtalk:v1.4.0
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 8060
              name: http
          resources:
            requests:
              cpu: 50m
              memory: 500Mi
            limits:
              cpu: 500m
              memory: 1000Mi
          volumeMounts:
            - mountPath: "/etc/prometheus-webhook-dingtalk/"
              name: webhook-config
            - mountPath: "/etc/prometheus-webhook-
dingtalk/templates/"
              name: webhook-template
      volumes:
```

```

- name: webhook-config
  secret:
    secretName: webhook-config
- name: webhook-template
  secret:
    secretName: webhook-template
tolerations:
- key: node-role.kubernetes.io/master
  operator: Exists
  effect: NoSchedule
---
apiVersion: v1
kind: Service
metadata:
  name: webhook-dingtalk
  namespace: monitoring
spec:
  selector:
    app: webhook-dingtalk
  ports:
  - name: hook
    port: 8060
    targetPort: http

```

找到Alertmanager的Secret更改进行部署部署

```

apiVersion: v1
kind: Secret
metadata:
  labels:
    alertmanager: main
    app.kubernetes.io/component: alert-router
    app.kubernetes.io/name: alertmanager
    app.kubernetes.io/part-of: kube-prometheus
    app.kubernetes.io/version: 0.21.0
  name: alertmanager-main
  namespace: monitoring
stringData:
  alertmanager.yaml: |-
    global:
      resolve_timeout: 1m
    route:
      group_by: ['alertname']
      group_wait: 10s

```

```

    group_interval: 10s
    repeat_interval: 1m
    receiver: 'webhook'
receivers:
- name: 'webhook'
  webhook_configs:
  - url: 'http://webhook-dingtalk:8060/dingtalk/webhook/send'
    send_resolved: true
inhibit_rules:
- source_match:
    severity: 'critical'
  target_match:
    severity: 'warning'
    equal: ['alertname',"target","job","instance"]
type: Opaque

```

如果根据标签进行分组报警到多个dingding群，比如标签为podtype: base的发送到基础服务群，普通的报警发送到技术群

```

apiVersion: v1
kind: Secret
metadata:
  labels:
    alertmanager: main
    app.kubernetes.io/component: alert-router
    app.kubernetes.io/name: alertmanager
    app.kubernetes.io/part-of: kube-prometheus
    app.kubernetes.io/version: 0.21.0
  name: alertmanager-main
  namespace: monitoring
stringData:
  alertmanager.yaml: |-
    global:
      resolve_timeout: 1m
    route:
      group_by: ['alertname']
      group_wait: 10s
      group_interval: 10s
      repeat_interval: 1m
      receiver: 'webhook'
      routes:
      - receiver: 'base'
        match:

```

```

        podtype: base #标签为podtype: base的发送到下面的base组, 也可以使用原有语句中的标签,
receivers:
- name: 'webhook'
  webhook_configs:
    #这里dingtalk/webhook/send后缀对应02webhook-secret.yaml中的config.yml.targets.webhook配置
    - url: 'http://webhook-dingtalk:8060/dingtalk/webhook/send'
      send_resolved: true
    #这里dingtalk/base/send后缀对应02webhook-secret.yaml中的config.yml.targets.base配置
    - name: 'base'
      webhook_configs:
        - url: 'http://webhook-dingtalk:8060/dingtalk/base/send'
          send_resolved: true
inhibit_rules:
- source_match:
    severity: 'critical'
  target_match:
    severity: 'warning'
  equal: ['alertname', 'target', 'job', 'instance']
type: Opaque

```

基于标签更改PrometheusRule规则增加上面说的自定义的podtype: base标签

```

apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  labels:
    prometheus: k8s
    role: alert-rules
  name: pods-rules
  namespace: monitoring
spec:
  groups:
  - name: pods.rules
    rules:
    - alert: PodMemory>90%
      annotations:
        description: 当前使用率 {{ printf "%.2f" $value }}%
        summary: 容器cpu使用率高于百分之90
      expr: |
        max by(pod) (rate(container_cpu_usage_seconds_total[3m]))
        / max by(pod) (container_spec_cpu_quota / 100000) * 100 > 90
      for: 1m

```

```
labels:
  severity: critical
  podtype: base #这里是自定义的, 会给查询出的语句自动增加一个
podtype: base标签。也可以使用原有语句中的标签
```

企业微信告警配置: 找到Alertmanager的Secret改为以下

```
apiVersion: v1
kind: Secret
metadata:
  labels:
    alertmanager: main
    app.kubernetes.io/component: alert-router
    app.kubernetes.io/name: alertmanager
    app.kubernetes.io/part-of: kube-prometheus
    app.kubernetes.io/version: 0.21.0
  name: alertmanager-main
  namespace: monitoring
stringData:
  alertmanager.yaml: |-
    global:
      resolve_timeout: 1m
    route:
      group_by: ['alertname']
      group_wait: 10s
      group_interval: 10s
      repeat_interval: 1m
      receiver: 'wechat'
    receivers:
    - name: 'wechat'
      wechat_configs:
      - corp_id: wwff383f5a86d412f5
        to_user: '@all'
        agent_id: 1000002
        api_secret: jPGLcXm4MvumPsEo44WGmfx5bJEUQTbEO80zSe25RDY
        send_resolved: true
    inhibit_rules:
    - source_match:
        severity: 'critical'
      target_match:
        severity: 'warning'
        equal: ['alertname', 'target', 'job', 'instance']
type: Opaque
```

