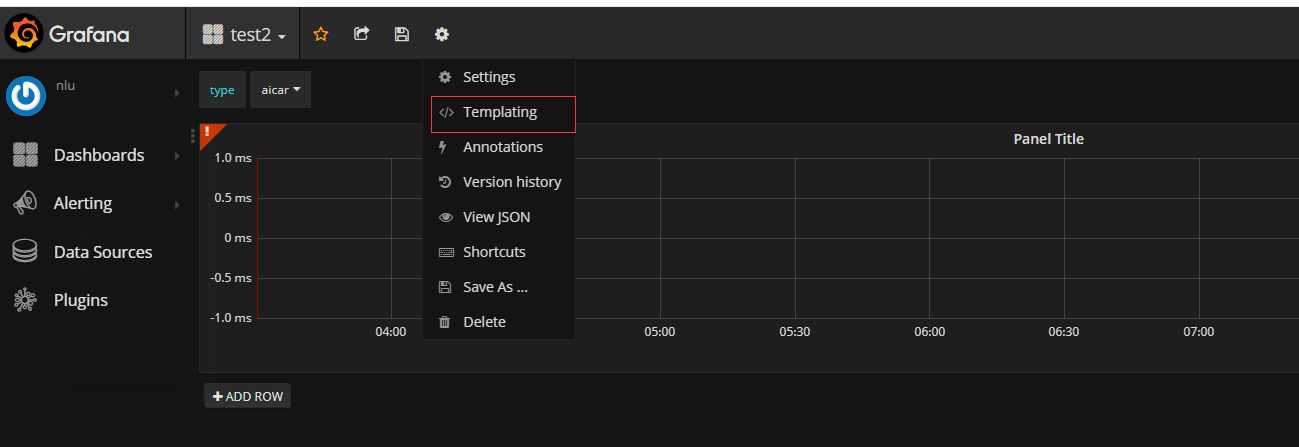
# 一、grafana查询中的变量templating

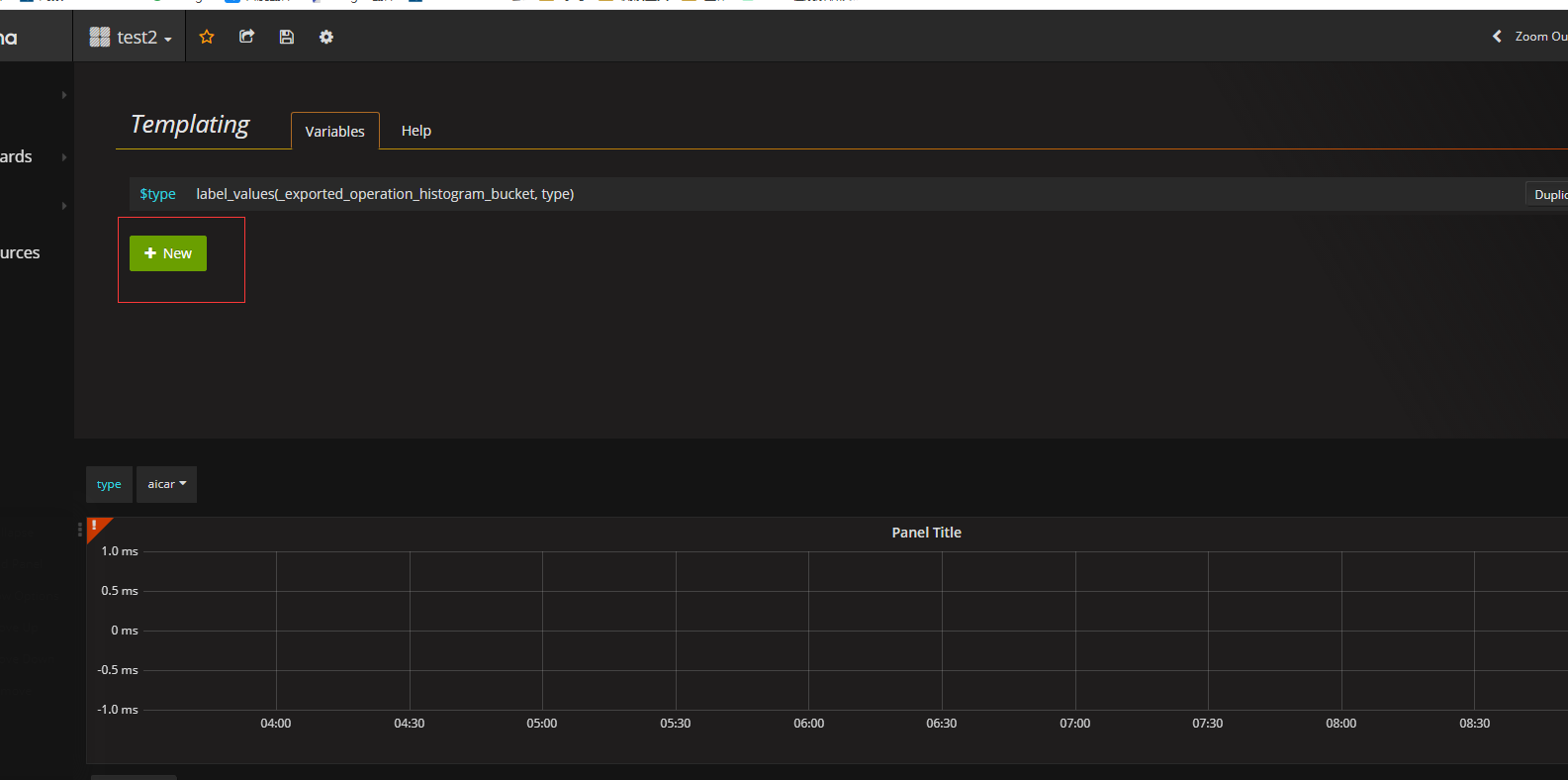
templating的变量其实就是事先帮我们查询好变量的值，然后在查询输入的时候事先帮我们把查询中的变量替换掉，从而实现一次配置，多台显示的目的。

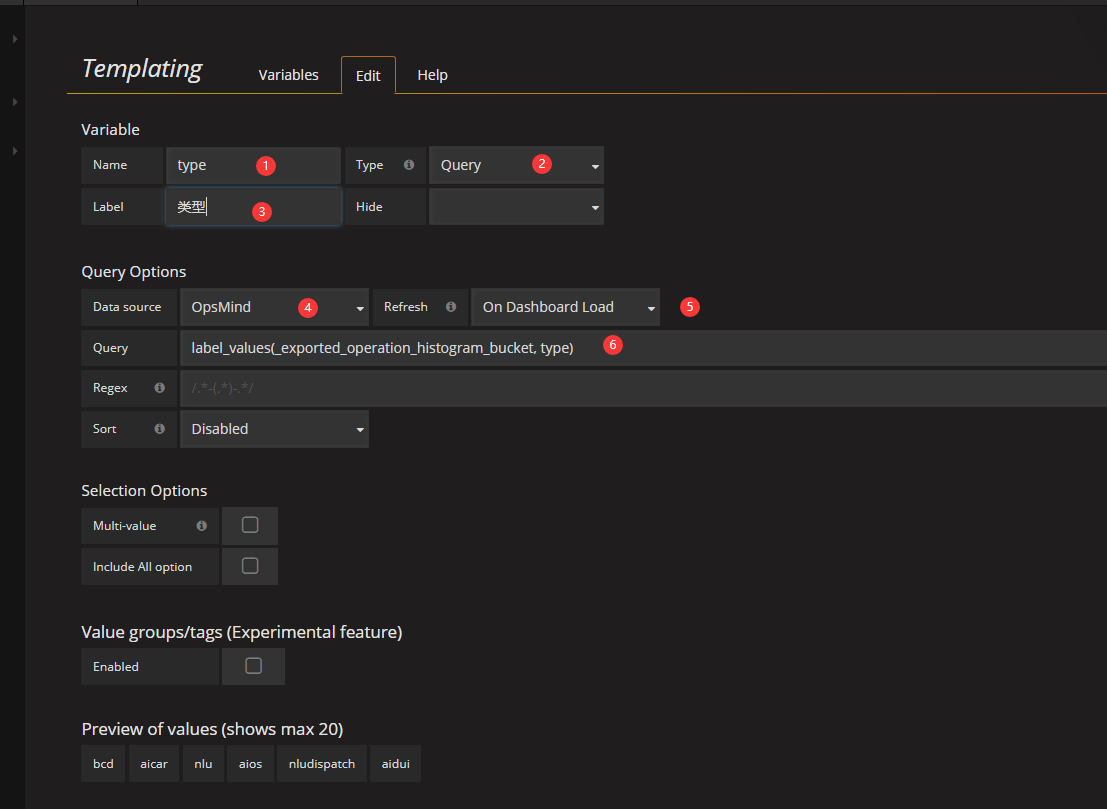
grafana有很多灵活的实现，甚至是不同的数据源也可以将查询融合到一起。

**a.首先我们打开templating**



**b.可以自己新加变量，但是这里我已经配置好了一个变量，type，我们来看它的具体内容**





1.name就是变量名啦，这里配置的是type，就是类型

2.type这里使用query，就是查询方式的变量

3.lable这是表示是对变量的描述(label\_values(\_exported\_operation\_histogram\_bucket, type))

4.datasource这里使用OpsMind

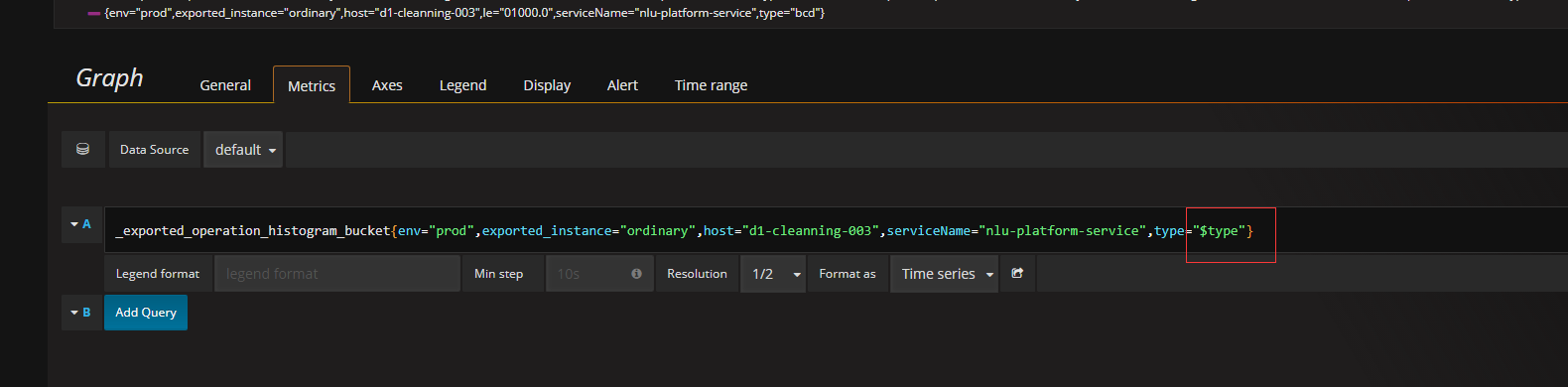
5.refresh选择在加载dashboard的时候，就是选择完主机就开始加载

6.query这里是prometheus语法的查询语句

**c.配置完变量之后就在dashboard的左上方出现了变量的选择下拉菜单**

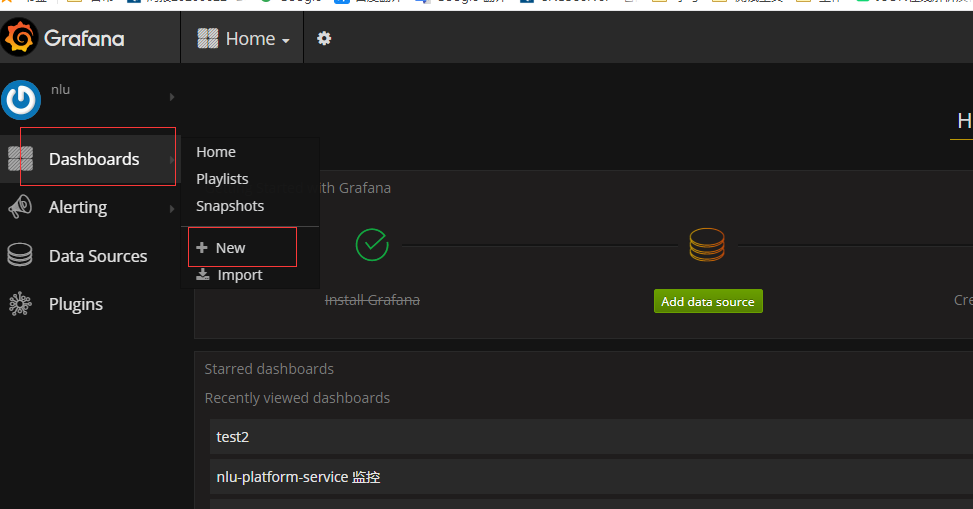


**d.修改查询语句，增加变量**

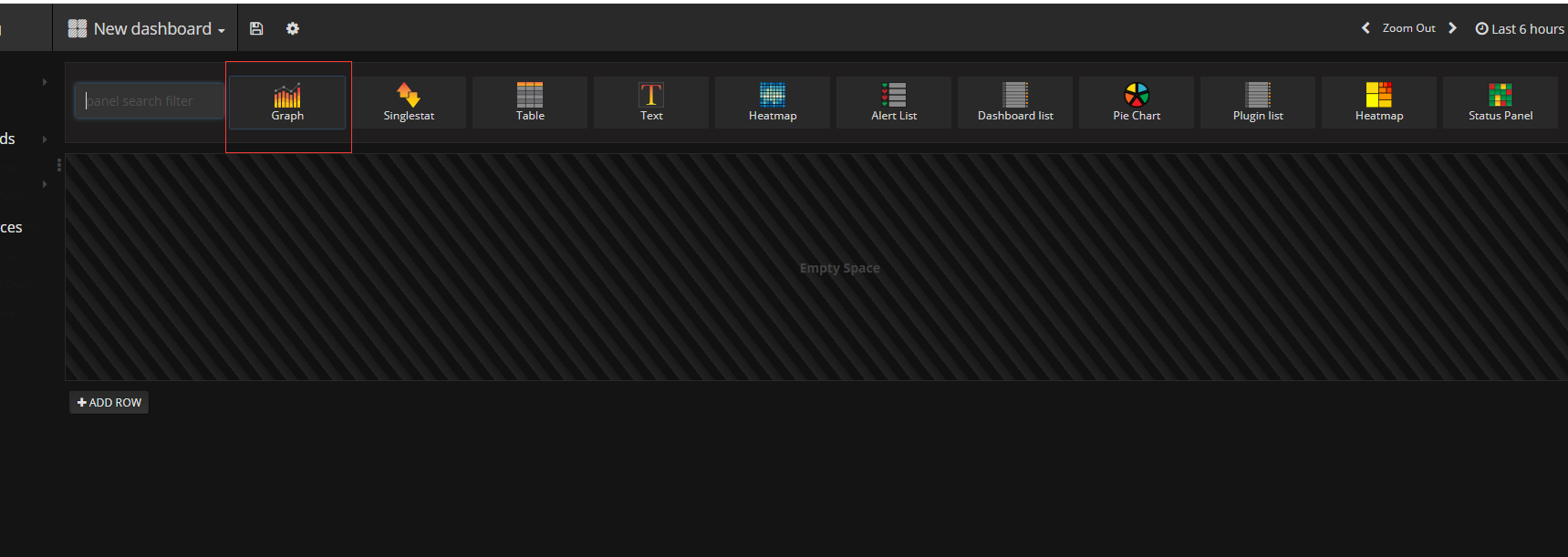


# 二、grafana创建监控面板

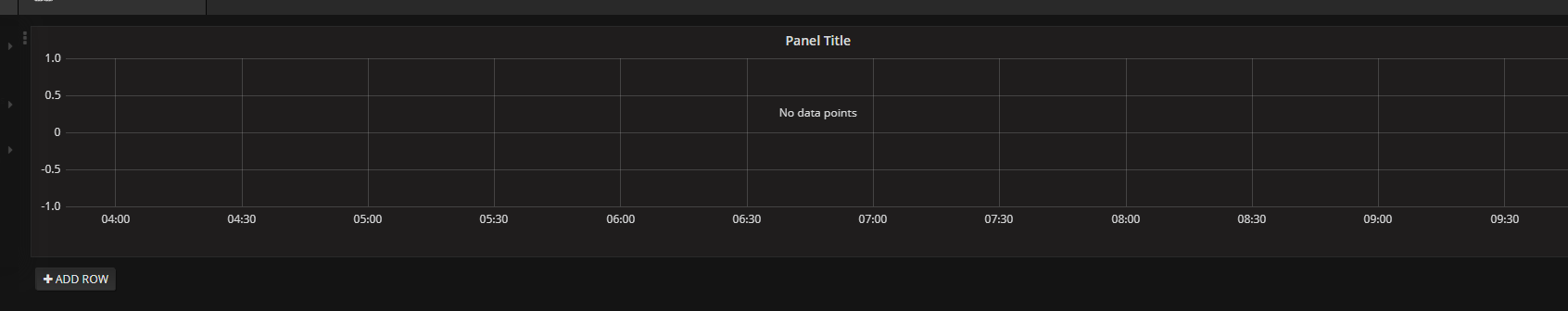
1.仪表盘创建



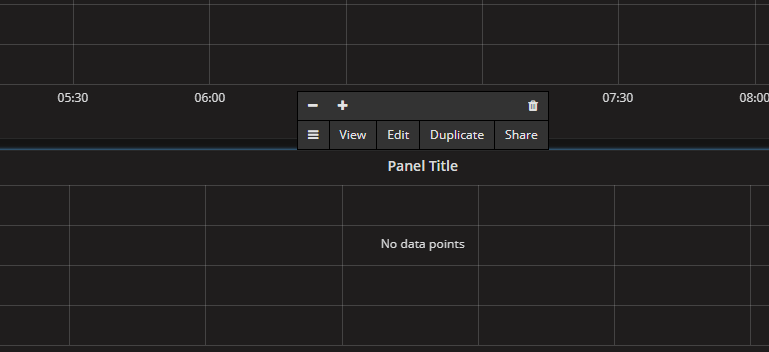
2.选择展示数据类型



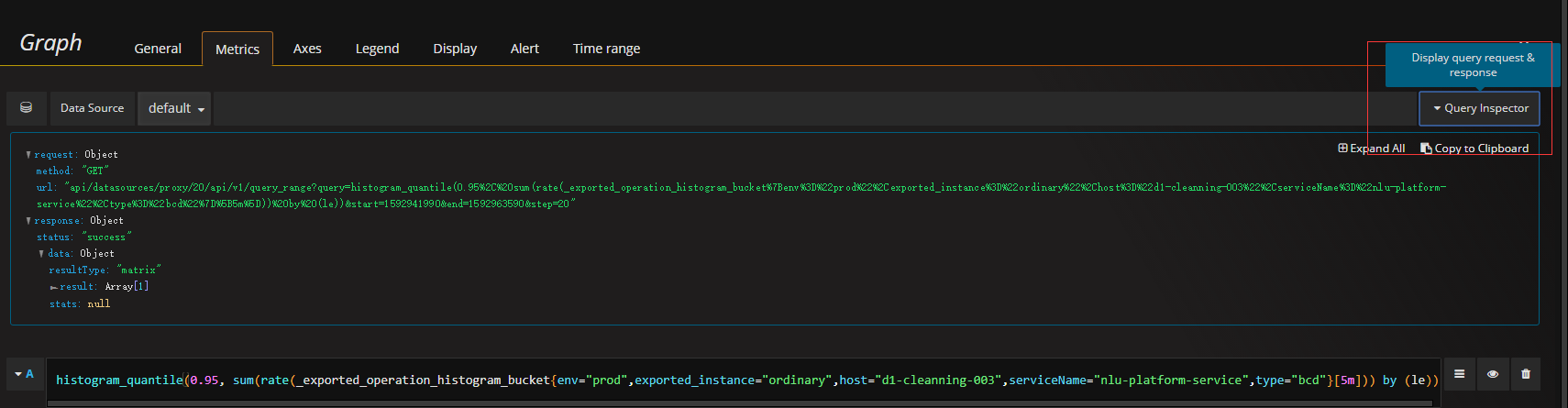
3.板块创建成功



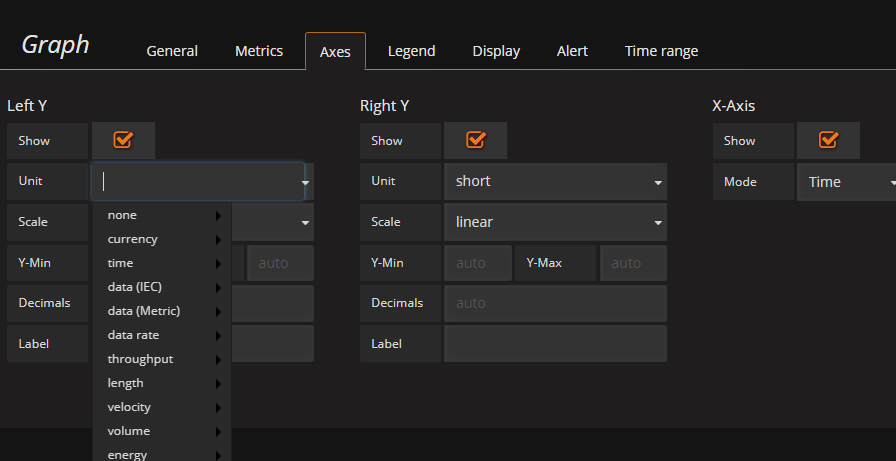
4.添加数据(鼠标点击panel title),点击Edit



5.添加查询语句(prometheus语法)



6.设置x坐标,y坐标



# 三、prometheus常用查询语法

## 1. 指标查询（瞬时向量查询）

通过指标名称和标签进行查询，可以查询该指标下的所有时间序列距离当前系统时间最新的值，无时间概念，所以查询的结果称为瞬时向量（instant vector），如下图所示：



而且可以看到查询到的多条时间序列都包含指定的标签。单独使用指标名称，相当于不使用标签进行过滤，同样独单使用标签查询也可以。

标签过滤支持使用=和!=两种完全匹配模式：

通过使用label=value可以选择那些标签满足表达式定义的时间序列；

反之使用label!=value则可以根据标签匹配排除时间序列；

除了使用完全匹配的方式对时间序列进行过滤以外，还支持使用正则表达式作为匹配条件，多个正则表达式之间使用|进行分离：

使用label=~regx表示选择符合正则表达式定义的时间序列；

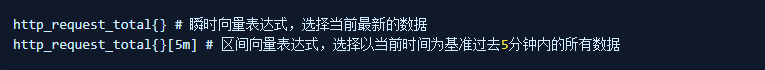
反之使用label=!~regx进行反向选择；

例如，如果想查询多个环境下的请求数统计，可以使用如下表达式：



## 2. 时间范围查询（区间向量查询）

​如果我们想过去一段时间范围内的样本数据时，我们则需要使用区间向量表达式。区间向量表达式和瞬时向量表达式之间的差异在于需要定义时间范围，通过时间范围选择器[]进行定义。



例如查询距离当前系统时间最近5分钟内的所有样本数据：可以看到结果中每个时间序列都有5个值并且@了不同的时间戳（因为该prometheus每分钟pull一次，所以5分钟有5个结果）。

除了使用m表示分钟以外，PromQL的时间范围选择器支持其它时间单位：

s - 秒

m - 分钟

h - 小时

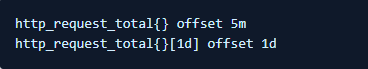
d - 天

w - 周

y - 年

## 3.时间位移

如果查询5分钟之前的最新数据可以使用位移操作,位移操作的关键字为offset



## 4. 操作符

除了能够方便的查询和过滤时间序列以外，还支持丰富的操作符，用户可以使用这些操作符进一步的对事件序列进行二次加工。这些操作符包括：数学运算符，逻辑运算符，布尔运算符等等。

数学运算符

操作数可以是一个常数，也可以是一个查询表达式，比如：



支持的所有数学运算符如下所示：

+ (加法)

- (减法)

\* (乘法)

/ (除法)

% (求余)

^ (幂运算)

布尔运算符

通过布尔运算对时间序列进行过滤，例如如下想查询node\_cpu\_seconds\_total{mode=“idle”} > 22000的时间序列，不大于的时间序列会被过滤掉。

目前，Prometheus支持以下布尔运算符如下：

== (相等)

!= (不相等)

> (大于)

< (小于)

>= (大于等于)

<= (小于

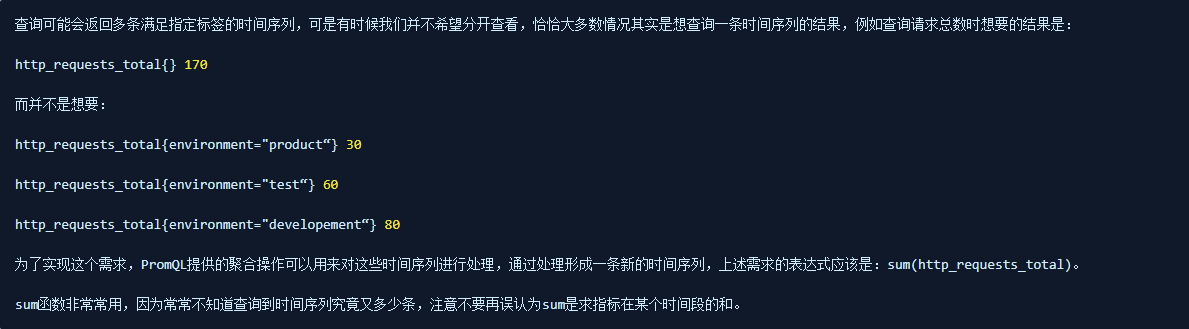
获取布尔运算结果

布尔运算的默认行为是对时间序列进行过滤。而有时候我们需要的是的运算结果。例如，只需要知道当前HTTP请求量是否>=1000，如果大于等于1000则返回1否则返回0。这时可以使用bool修饰符改变布尔运算的默认行为。 例如：



使用bool修饰符后，布尔运算不会对时间序列进行过滤，而是直接依次对各个样本数据进行比较，结果是0或者1。从而形成一条新的时间序列。

## 5. 聚合函数



常用的聚合函数

sum：求和。

min：最小值。

max：最大值

avg：平均值

stddev：标准差

stdvar：方差

count：元素个数

count\_values：等于某值的元素个数

bottomk：最小的 k 个元素

topk：最大的 k 个元素

quantile：分位数

部分聚合

有时候，聚合并不想完全聚合，想根据某个标签进行区分时候，可以使用by进行拆分，比如监控每个cpu累计的空闲时间：sum(node\_cpu\_seconds\_total{mode=“idle”} )by (cpu)，并设置了时间序列的名称模式为：cpu-{{cpu}}