Типы данных в PromQL

В запросах Prometheus есть 3 типа данных:

Instant vector – вектор содержит в себе все значения метрики по запрашиваемой метке времени.

Range vectors – возвращает все вектора за указанный период времени. Это позволяет увидеть изменение метрики во времени.

Scalar – простое числовое значение с плавающей точкой.

Для некоторых запросов есть ограничения для типа данных. Если в запросе могут быть использованы только определенные типы данных, это будет указано в описании запроса.

Операторы PromQL

Математические операторы:

- +

Этот оператор производит сложение. Оператор сложения возможно использовать только с instant vector и scalar.

Пример: 1+2 результат будет 3.

- -

Этот оператор производит вычитание. Оператор вычитания возможно использовать только с instant vector и scalar.

Пример: 3-2 результат будет 1.

- *

Этот оператор производит умножение. Оператор умножения возможно использовать только с instant vector и scalar.

Пример: 2*2 результат будет 4.

- /

Этот оператор производит деление. Оператор деления возможно использовать только с instant vector и scalar.

Пример: 8/4 результат будет 2.

- %

Этот оператор возвращает модуль исходных данных. Оператор % возможно использовать только с Iinstant vector и scalar.

Пример: 13 % 5 результат будет 3

- ^

Этот оператор производит возведение в степень. Оператор возведения в степень возможно использовать только с instant vector и scalar.

Пример: 2 ^ 10 результат будет 1024

Операторы сравнения:

- == – равно
- != – не равно
- > – больше
- < – меньше
- >= – больше или равно
- <= – меньше или равно

Сравнение возможно только над scalar и instant vector. Оператор применяется к каждому значению в исходном векторе, и все элементы данных, для которых не выполняется условие сравнения, исключаются из результирующего вектора.

Если указан модификатор bool, то в результирующем векторе данные, которые удовлетворяют условию сравнения, будут иметь значение 1, а не удовлетворяющие 0.

NB! Сравнение двух scalar возможно только с модификатором bool.

Чтобы лучше понять логику работы операторов сравнения, выполните последовательно следующие запросы и сравните результаты:

node_cpu_seconds_total
node_cpu_seconds_total > 3
node_cpu_seconds_total > bool 3

Логические операторы:

- and – логическое И
- or – логическое ИЛИ

- unless – дополнение

Логические операторы возможны только между instant vector.

vector1 and vector2 приводят к вектору, состоящему из элементов vector1 , для которых в vector2 есть элементы с точно совпадающими наборами меток. Другие элементы отброшены. Название и значения метрики переносятся из левого бокового вектора.

vector1 or vector2 приводит к вектору, который содержит все исходные элементы (наборы меток + значения) vector1 и дополнительно все элементы vector2 , которые не имеют совпадающих наборов vector1 .

vector1 unless vector2 приводит к вектору, состоящему из элементов vector1 , для которых нет элементов в vector2 с точно совпадающими наборами меток. Все совпадающие элементы в обоих векторах отбрасываются.

Чтобы лучше понять логику работы операторов сравнения, выполните последовательно следующие запросы и сравните результаты:

node_load1 and node_cpu_seconds_total
node_load1 or node_cpu_seconds_total
node_load1 unless node_cpu_seconds_total

Сопоставления векторов

Сопоставление векторов производит поиск одинаковых элементов в правом векторе для каждой записи левого вектора. Есть 2 типа сопоставления: один к одному и один ко многим / многие к одному.

Сопоставление один к одному.

При сопоставление один к одному для всех элементов из правого вектора ищется элемент, имеющий точно такой же набор меток (учитываются как имена метки, так и их значения), и над такими парами производится операция.

Существует 2 модификатора для сравнения:

- on – задает, какие labels необходимо учитывать при сопоставлении.
- ignoring – задает, какие labels должны быть исключены в процессе сопоставления.

Чтобы лучше понять данный механизм, попробуйте выполнить запрос:

node_cpu_seconds_total{instance="server1:9100",job="static_config"} + on(cpu, mode)
node_cpu_seconds_total{instance="server2:9100",job="static_config"}

Обратите внимание: в запросе необходимо изменить значения server1 и server2 на ip адреса серверов.

В результате данного запроса вы получите суммарное потребление cpu по двум серверам.

Сопоставление один ко многим / многие к одному: в этом случае одному элементу правого вектора может соответствовать несколько элементов второго вектора. Тогда необходимо явно указать, какой вектор является главным. Это можно сделать с помощью ключевых слов **group_left** и **group_right.**

Операторы агрегации:

- sum – сумма
- min – минимальное значение
- max – максимальное значение
- avg – среднее значение
- stddev – стандартное отклонение
- stdvar – стандартная дисперсия
- count – количество элементов в векторе
- count_values – количество элементов с одинаковым значением.

Данные операторы могут использоваться с модификаторами **by** и **without**. С помощью модификатора by задается список labels, которые будут учитываться, а модификатор without задает список labels, которые учитываться не будут. Указание данного модификатора допускается как до, так и после запроса.

operator ([parameter,] <vector expression>) [without|by (<label list>)]

Например, чтобы получить суммарное потребление cpu по всем инстансам в user mode, выполните следующий запрос:

sum(node_cpu_seconds_total{mode="user"}) by (cpu)

Чтобы получить суммарное потребление процессора в user mode по всем ядрам, выполните запрос:

sum without (cpu) (node_cpu_seconds_total{mode="user"})

Приоритет бинарных операторов

Бинарные операторы имеют следующий приоритет:

1. ^

2. * , / , %
3. + , -
4. == != , <= , < , >= , >
5. and , unless
6. or