Одной из самых часто используемых команд docker-а является команда docker stats. Так же как docker ps. То есть, что она делает. Она собирает все контейнеры и выводит их статистику по потреблению памяти, процессора и сети. Соответственно, эта команда стримит, то есть эта команда постоянно висит до тех пор, пока вы ее не отмените. И мы можем воспроизвести нечто подобное с командой ContainerStats. Она очень простая, она в себя принимает context, принимает ID-контейнеры, для которого вы хотите стримить, значит – для которого вы хотите получить значение ресурсов. И как раз позволяет вам определить, хотите ли вы стримить или хотите вы просто один раз считать значения. И как бы, остановиться на этом. И в ответ вы будете получать, в любом случае, json-ы в переменный ContainerStats. То есть здесь у вас будет выбор. И соответственно, вы будете получать метрики, из которых вы можете построить аналогичную абсолютно табличку. Давайте посмотрим на ответ этой функции. И какие метрики мы можем с ней построить.

Итак, я запустил мою команду stats, вместе со stream: false, то есть она мне выдала JSON и вышла. Если мы ее запустим точно так же со stream: true, то мы здесь увидим, соответственно, вот такой бесконечный…, бесконечный поток данных, которые…, ну соответственно, они разделены, естественно. Там можно подождать, какие-то данные выбрать, но оно постоянно будет стримиться. То есть мы можем это использовать, чтобы обрабатывать этот поток. И соответственно, выводить оттуда данные.

Если мы посмотрим на JSON, который нам выдается, то мы видим здесь следующее: read - когда было сделано чтение, preread – когда мы начали чтение, это нужно для правильного расчета времени компьютера, то есть нам нужно знать когда мы начали, когда мы закончили. Соответственно, pids, то есть Process IDentifiers запущенные.

И переходим к интересному моменту cpu_stats. То есть что это такое. Тут мы можем видеть, что cpu_usage есть, total_usage, значит, percpu_usage, system_cpu_usage, online_cpus. На моей машине установлено 8 ядер, значит у меня 8 процессоров по мнению docker-a. И есть ли у меня какой-то throttling. Первый раз, когда я это увидел, я ничего не понял. Я был несколько в ступоре, но, в принципе, здесь довольно просто. Это количество тактов процессора использованных, против количества тактов процессора доступных. То есть в общем виде вы можете просто их поделить друг на друга, умножить на 100 и получить процент.

В чем здесь фишка. Во-первых, у вас есть еще разбивка по самому ядру процессора, а во-вторых, у вас есть precpu и percpu. То есть то, что было перед запросом и то, что как бы во время запроса. Для того, чтобы как раз определить вот эту вот дельту. Потому что на запрос вы тоже тратите какое-то время, какие-то такты процессора, они включаются в общую статистику.

По memory здесь все проще. Здесь у нас, соответственно, абсолютно обычные байты. И вы все это спокойно можете…, этот лимит посчитать по памяти. Здесь все довольно просто.

С процессором, опять же, когда я про это забываю, как считается, я захожу в исходники docker-клиента, то есть того самого который делает docker stats, и смотрю, как они там это все считают. То есть видно, что тут у нас TotalUsage есть – это cpuDelta, и SystemUsage – это systemDelta. То есть мы их просто делим друг на друга и умножаем на 100 процентов. И соответственно, получаем какой-то процент. Но здесь, опять же, есть еще previousCPU, то есть вот этот вот как раз PreCPUStats, то есть мы таким образом можем четко определить, когда мы, соответственно, начали, когда мы…, какие мы CPU используем системы, а какие-то нет. И получить плюс-минус реальную картину. Я обычно не

заморачиваюсь и прямо считаю их напрямую. То есть работает это вот примерно как-то..., каким-то таким образом.