

Тестируй сам с Yandex.Tank

Эффективное нагрузочное тестирование для тех, у кого мало времени

А кто я?



Ильи Иванкин.

– Head of backend, 2022 - Present.

Past:

– Lead SE. EPAM

– Tech lead. SFT





О чем доклад?

История одного проекта

Требования или SLA/SLO/SLI

Тесты и как их запускать

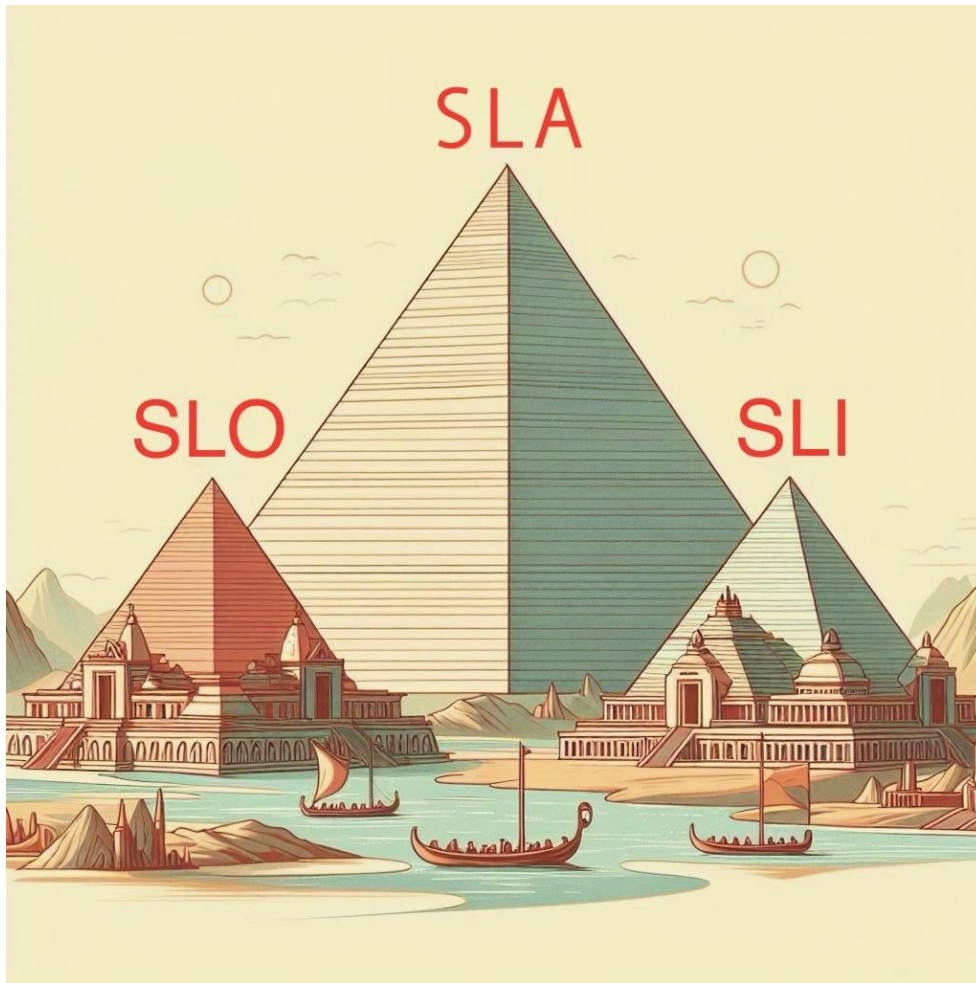
Инспекция сервиса и методов

Итоги

История одного проекта



К нам в команду пришел заказчик и поставил нам классическую задачу - разработать сервис, на который ожидается приличная нагрузка даже в MVP.





Service Level Indicator (SLI)

История одного проекта



- **RPS**
- **Error rate**
- **Latency**
- **Availability**

История одного проекта



Service Level Indicator (SLI)

Service Level Objective (SLO)

История одного проекта



- **RPS - 500**
- **Error late - 5%**
- **Latency - 1s**
- **Availability - 99.99**

Все еще теория

А как измерить основные показатели? Среднее время или процентиль?



0.700, 0.720, 0.680, 0.660, 0.740, 0.750,
0.730, 0.670, 0.710, 0.200, 0.150, 0.300,
0.350, 0.400, 0.450, 0.500, 0.550, 0.600,
0.250, 0.320, 0.380, 0.420, 0.490, 0.530,
0.580, 0.620, 0.310, 0.370, 0.440, 0.510,
0.560, 0.610, 0.290, 0.340, 0.390

Среднее арифметическое ≈ 0.429

Все еще теория

Медиана или 50 процентиль



0.150, 0.200, 0.250, 0.290, 0.300,
0.310, 0.320, 0.340, 0.350, 0.370,
0.380, 0.390, 0.400, 0.420, 0.440,
0.450, 0.490, 0.500, 0.510, 0.530,
0.550, 0.560, 0.580, 0.600, 0.610,
0.620, 0.660, 0.670, 0.680, 0.700,
0.710, 0.720, 0.730, 0.740, 0.750

18-е число в упорядоченном списке: 0.510.

Медиана

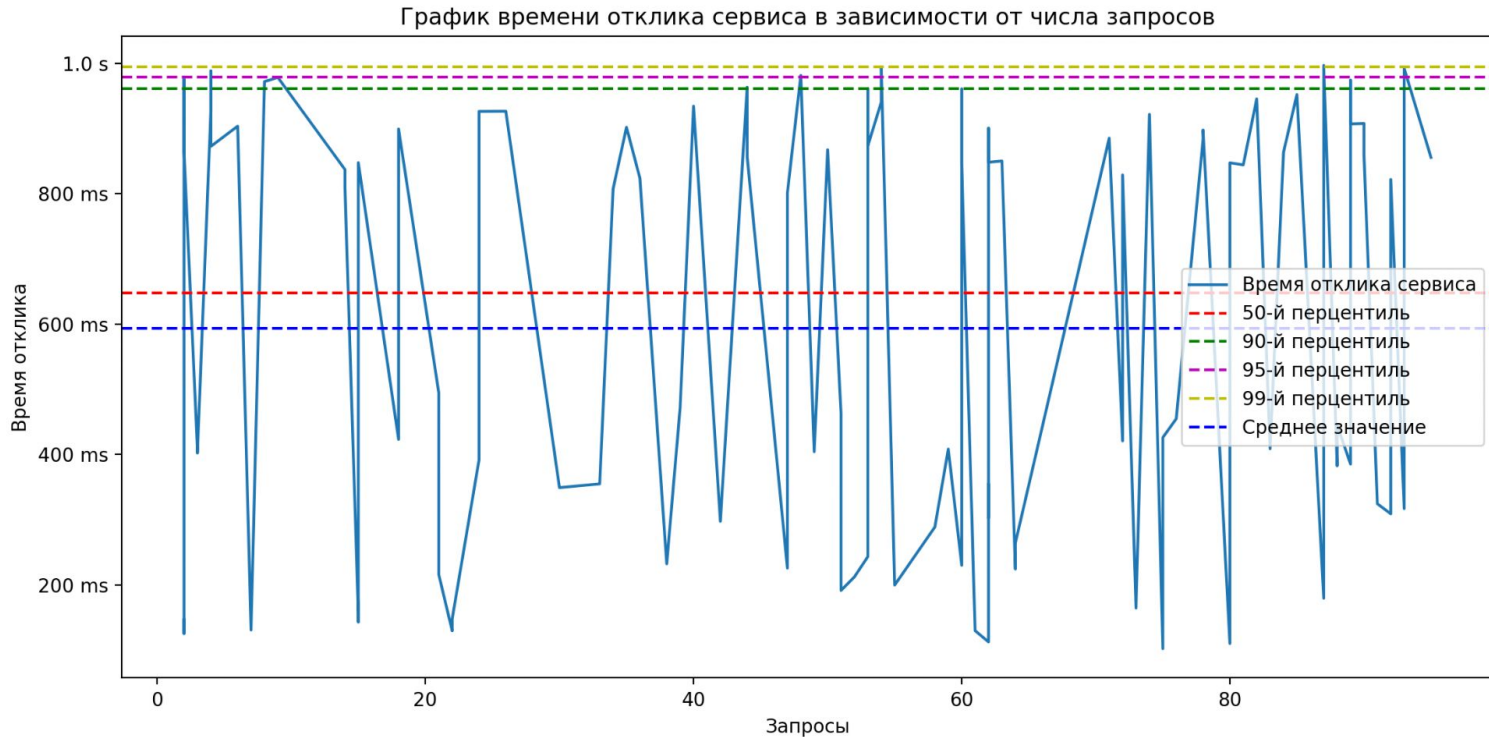


Медиана разделяет набор данных на две равные части и не зависит от всех значений в наборе, в то время как среднее значение рассчитывается на основе всех чисел в наборе, что делает его чрезвычайно чувствительным к аномально высоким или низким значениям.

Эти выбросы могут существенно исказить среднее значение, делая его менее представительным для описания центральной тенденции набора данных.

Все еще теория

А как измерить основные показатели? Среднее время или процентиль?



История одного проекта



Service Level Indicator (SLI)

Service Level Objective (SLO)

Service Level Agreement (SLA)



CRM



Наши соглашения

Стартовый RPS: **300**



SLA:

- 99.9% доступности сервиса в течение каждого месяца.

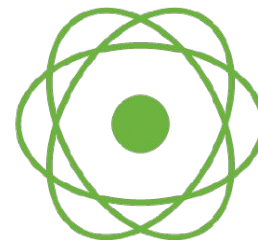
SLI:

- **97% всех запросов имеют код 2**, не более 3% error rate**

SLO:

- find 99% - 300 мс, по 95% за 500 мс.
- find top 10: 99% - 100ms, 95% - 200ms
- save: 99% - 2s, 95% - 1s

История одного проекта



R2DBC



Что добавить в сервис, чтобы собрать метрики?

желательно из коробки и ничего больше не трогать



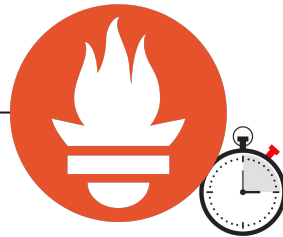
Как работает эта связка?



```
- job_name: stock-service-1
  metrics_path: /actuator/prometheus
  static_configs:
    - targets: ['stock-service-1:8084']
```



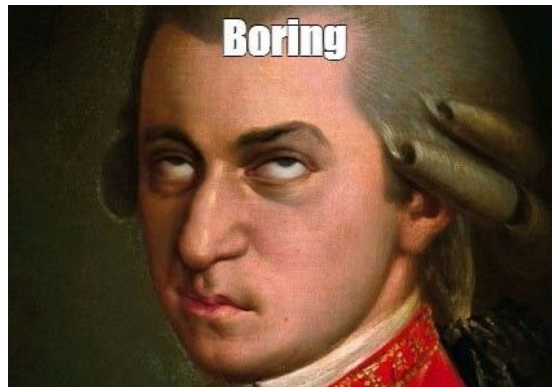
/actuator/prometheus



http://localhost:8084/actuator/prometheus

Механики:

- имитируем пользователей
- измерение производительности и времени отклика
- анализ результатов





Механики и цели

- имитируем пользователей - 300 users
- измерение производительности и времени отклика - 500ms поиск
- анализ результатов - куда смотреть?

Требования есть, сервис есть, как тестировать?



Наш сервис - это HTTP сервис. Наша задача пробить сервис через внешний URL и выйти на те цифры что, от нас ждут.

Встречайте: Яндекс.Танк - инструмент, благодаря которому можно сэкономить кучу времени и не писать свое решение, особенно если времени у вас нет!



Яндекс.Танк



"Яндекс.Танк" - это инструмент для проведения нагрузочного тестирования веб-приложений и служб.

Компоненты "Яндекс.Танка":

1. Core (Ядро):
2. Load Agents (Агенты нагрузки):
3. Configuration (Конфигурация)
4. Scenarios (Сценарии)
5. Metrics (Метрики) и отчеты

Яндекс.Танк

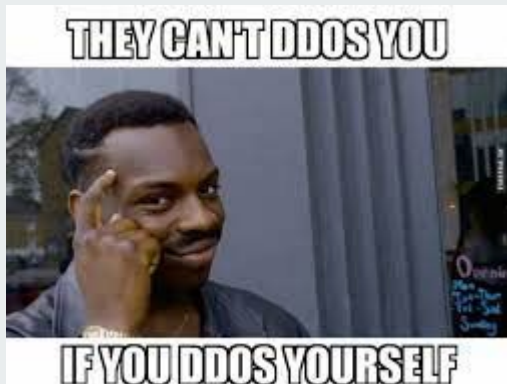


Соберем базовый конфиг для
нашего первого запроса?

```
phantom:  
  address: "localhost"  
  port: "8085"  
  load_profile:  
    load_type: rps  
    schedule: const(1500, 60s)  
writelog: all  
ssl: false  
connection_test: true  
uris:  
  - "/api/v1/stocks"
```


Яндекс.Танк

Добавим автостоп! скажем нет DDOS.



autostop:

autostop:

- `time(1s,10s)` # *if request average > 1s*
- `http(5xx,3%,1s)` # *if 500 errors > 1s*
- `http(4xx,3%,1s)` # *if 400 > 25%*

Яндекс.Танк



Можно фулл?



```
phantom:  
  address: "localhost"  
  port: "8085"  
  load_profile:  
    load_type: rps  
    schedule: const(1500, 60s)  
  writelog: all  
  ssl: false  
  connection_test: true  
  uris:  
    - "/api/v1/stocks"  
telegraf:  
  enabled: false  
autostop:  
  autostop:  
    - time(1s,10s) # if request average > 1s  
    - http(5xx,100%,1s) # if 500 errors > 1s  
    - http(4xx,25%,10s) # if 400 > 25%
```

Сервис есть, конфиг есть, а как запустить?



```
docker run \  
  -v $(pwd):/var/loadtest \  
  --net="host" \  
  -it direvius/yandex-tank -c find-load.yml
```



Круто, а куда смотреть?





После запуска все будет в консоле!

```
Terminal find-tank.sh x
Data delay: 4s, RPS: 1,500
Percentiles (all/last 1m/last), ms:
100.0% < 253.0 253.0 2.1
99.5% < 15.0 15.0 1.5
99.0% < 4.4 4.4 1.4
95.0% < 1.2 1.2 1.1
90.0% < 1.0 1.0 1.0
85.0% < 0.9 0.9 0.9
80.0% < 0.9 0.9 0.8

HTTP codes:
. 52,608 +1,500 100.00% : 200 OK

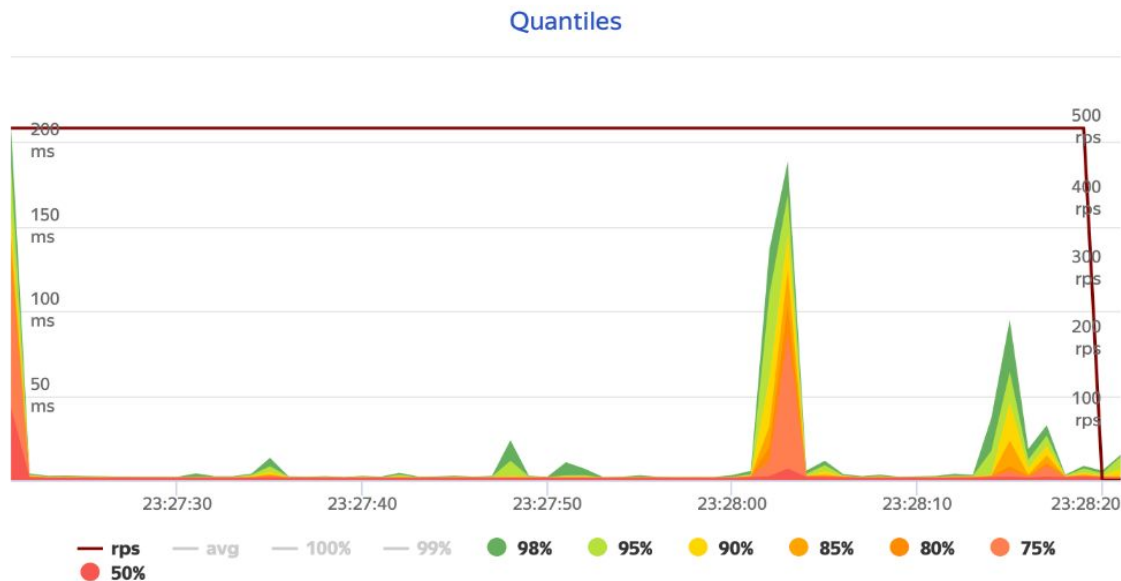
Net codes:
. 52,608 +1,500 100.00% : 0 Success

Average Sizes (all/last), bytes:
. Request: 31.0 / 31.0

Duration: 0:00:39 ETA: 0:00:21
Author: root
Job: 680531 find stocks
Task:
Web: https://overload.yandex.net/680531
Hosts: docker-desktop => localhost:8085
Ammo:
Count: 90000
```



А можно ли посмотреть на графики?



Highcharts.com

15 Feb 18:45:37



15 Feb 18:45:37

15 Feb 18:45:52

15 Feb 18:46:07

Overall const(1500, 60s)

Overview

Extended analysis

Distributions

Tables

Overall



Cumulative quantiles

	Quantile
99%	4.970
98%	2.250
95%	1.320
90%	1.070
85%	0.960
80%	0.890
75%	0.830
50%	0.650



Спойлер - конфиг мы сильно не поменяем, а вот чуток кода пописать придется!

У Яндекс.Танк есть в документации пример генератора патронов для танка, который написан на питоне и его достаточно просто адаптировать под нас.

а POST запрос?

```
def generate_json():
    body = {
        "name": "content",
        "price": 1,
        "description": "description"
    }
    url = "/api/v1/stock"
    h = headers + "Content-type: application/json"
    s1 = json.dumps(body)
    ammo = make_ammo(method, url, h, case, s1)
    sys.stdout.write(ammo)
    f2 = open("ammo/ammo-json.txt", "w")
    f2.write(ammo)
```



а POST запрос?



212

POST /api/v1/stock HTTP/1.1

Host: test.com

User-Agent: tank

Accept: */*


Connection: Close

Content-type: application/json

Content-Length: 61

```
{"name": "content", "price": 1, "description": "description"}
```

Яндекс.Танк и POST запрос



Создаем новый конфиг или модифицируем старый.

Главное указать какой тип патрона и сам патрон.

Удаляем URI, которые были до.

Остальное можно не трогать.

```
phantom:  
  address: "localhost"  
  port: "8085"  
ammo_type: phantom  
ammofile: ammo-json.txt  
load_profile:  
  load_type: rps  
  schedule: const(1500, 60s)
```


Cumulative quantiles

	Quantile
99%	1.880
98%	1.320
95%	1.110
90%	0.990
85%	0.900
80%	0.840
75%	0.800
50%	0.670

Еще чуток метрик



Поиск



```
phantom:  
  address: "localhost"  
  port: "8085"  
  load_profile:  
    load_type: rps  
    schedule: const(1500, 60s)  
  writelog: all  
  ssl: false  
  connection_test: true  
  uris:  
    - "/api/v1/stock?symbol=OMCL"  
    - "/api/v1/stock?symbol=TLGYW"
```

А что с поиском?

```
Terminal  find-one-tank.sh ×  build.sh ×  +  ▾
Percentiles (all/last 1m/last), ms:  . HTTP codes:
100.0% <  3,100.0  3,100.0  3,099.3  .  1,287 +339  100.00% : 200 OK
 99.5% <  3,080.0  3,080.0  3,099.3  .
 99.0% <  3,070.0  3,070.0  3,089.6  . Net codes:
 95.0% <  3,040.0  3,040.0  3,066.6  .  1,287 +339  100.00% :  0 Success
 90.0% <  3,000.0  3,000.0  3,049.3  .
 85.0% <  2,885.0  2,885.0  3,039.9  . Average Sizes (all/last), bytes:
 80.0% <  2,735.0  2,735.0  3,032.1  . Request: 42.4 / 42.4
19:14:42 [WARNING] Autostop criterion requested test stop: time(500ms,5s)
19:14:42 [WARNING] Autostop criterion requested test stop: Average response time higher than 500ms for 5s, since 170802
19:14:42 [INFO] Finishing test...
19:14:42 [INFO] Stopping load generator and aggregator
```

А что с поиском?

```
Terminal  find-one-tank.sh x  build.sh x  +  v
Percentiles (all/last 1m/last), ms:  . HTTP codes:
100.0% < 3,100.0  3,100.0  3,099.3  . 1,287 +339 100.00% : 200 OK
 99.5% < 3,080.0  3,080.0  3,099.3  .
 99.0% < 3,070.0  3,070.0  3,089.6  . Net codes:
 95.0% < 3,040.0  3,040.0  3,066.6  . 1,287 +339 100.00% : 0 Success
 90.0% < 3,000.0  3,000.0  3,049.3  .
 85.0% < 2,885.0  2,885.0  3,039.9  . Average Sizes (all/last), bytes:
 80.0% < 2,735.0  2,735.0  3,032.1  . Request: 42.4 / 42.4
19:14:42 [WARNING] Autostop criterion requested test stop: time(500ms,5s)
19:14:42 [WARNING] Autostop criterion requested test stop: Average response time higher than 500ms for 5s, since 170802
19:14:42 [INFO] Finishing test...
19:14:42 [INFO] Stopping load generator and aggregator
```

Индекс

create index idx_symbol on **stocks** (symbol)



После добавления:



```

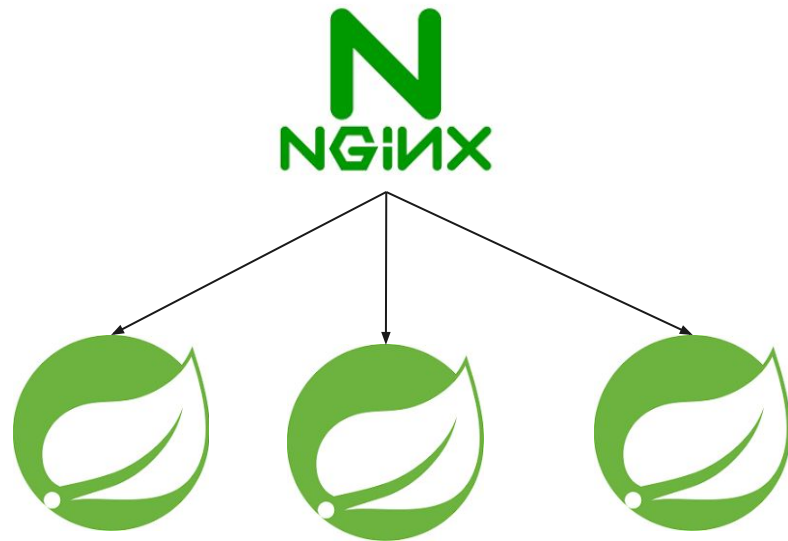
Terminal    find-one-tank.sh ×  build.sh ×  +  -
Data delay: 4s, RPS: 1,502 ████████████████████████████████████████████████████████████████████████████

Percentiles (all/last 1m/last), ms:  . HTTP codes:
100.0% <  338.0  338.0  11.5      .  83,774 +1,502  100.00% : 200 OK
 99.5% <    7.5    7.5    2.3      .
 99.0% <    2.7    2.7    1.6      . Net codes:
 95.0% <    1.1    1.1    1.1      .  83,774 +1,502  100.00% :  0 Success
 90.0% <    0.9    0.9    1.0      .
 85.0% <    0.9    0.9    0.9      . Average Sizes (all/last), bytes:
 80.0% <    0.8    0.8    0.8      . Request: 42.4 / 42.4 ██████████████████████████████████████████████████████████████████████████████████
19:19:42 [INFO] Phantom done its work with exit code: 0
19:19:42 [INFO] Finishing test...
  
```



Добавим балансёр

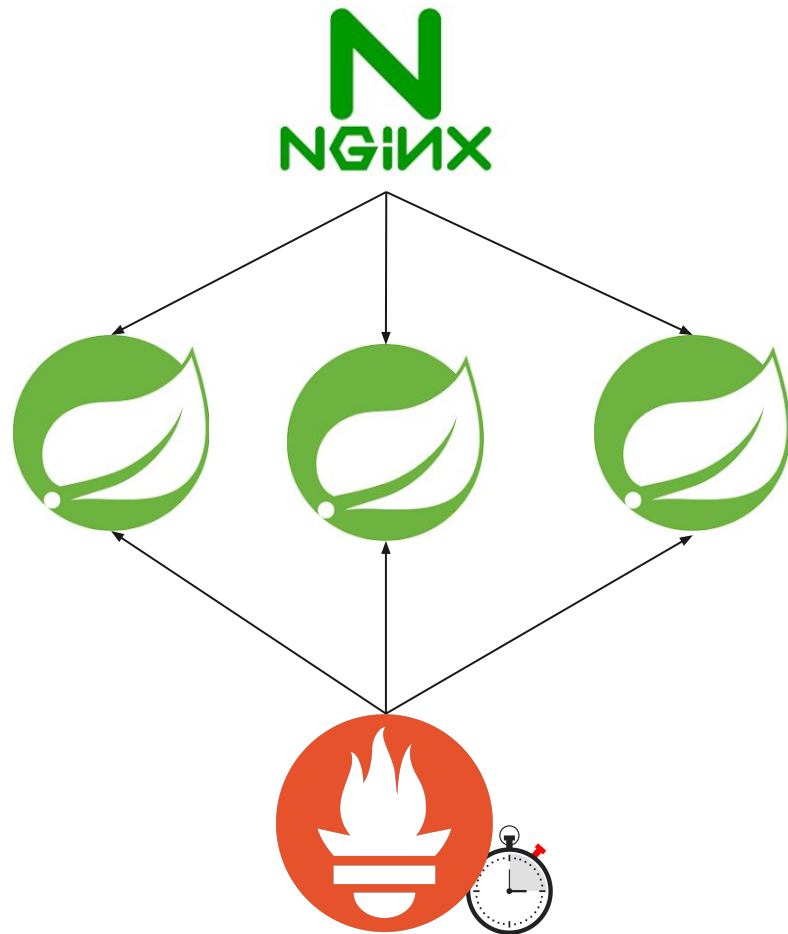
а почему бы не nginx?





Добавим балансир

Добавим в пром еще метрик





Но есть один нюанс...

```
Data delay: - s, RPS: 0
```

```
Percentiles (all/last 1m/last), ms: . HTTP codes:
                                     .
                                     .
                                     . Net codes:
                                     .
                                     .
                                     . Average Sizes (all/last), bytes:
                                     .
```

```
22:39:56 [WARNING] Autostop criterion requested test stop: http(5xx,3%,1s)
```

```
22:39:56 [WARNING] Autostop criterion requested test stop: 5xx codes count higher than 3.0% for 1s, since 1709851192
```

А это не он

```
[07/Mar/2024:20:29:09 +0000] "GET /api/v1/stock?symbol=OCUP HTTP/1.0" 502 157 "-" "-"
[07/Mar/2024:20:29:09 +0000] "GET /api/v1/stock?symbol=AACG HTTP/1.0" 502 157 "-" "-"
[07/Mar/2024:20:29:09 +0000] "GET /api/v1/stock?symbol=NYCB HTTP/1.0" 502 157 "-" "-"
[07/Mar/2024:20:29:09 +0000] "GET /api/v1/stock?symbol=OCGIO HTTP/1.0" 502 157 "-" "-"
[07/Mar/2024:20:29:09 +0000] "GET /api/v1/stock?symbol=TLGYW HTTP/1.0" 502 157 "-" "-"
[07/Mar/2024:20:29:09 +0000] "GET /api/v1/stock?symbol=ACONW HTTP/1.0" 502 157 "-" "-"
[07/Mar/2024:20:29:09 +0000] "GET /api/v1/stock?symbol=OABI HTTP/1.0" 502 157 "-" "-"
[07/Mar/2024:20:29:09 +0000] "GET /api/v1/stock?symbol=OCUP HTTP/1.0" 502 157 "-" "-"
[07/Mar/2024:20:29:09 +0000] "GET /api/v1/stock?symbol=TLGYW HTTP/1.0" 502 157 "-" "-"
[07/Mar/2024:20:29:09 +0000] "GET /api/v1/stock?symbol=ADNWW HTTP/1.0" 502 157 "-" "-"
[07/Mar/2024:20:29:09 +0000] "GET /api/v1/stock?symbol=OABI HTTP/1.0" 502 157 "-" "-"
[07/Mar/2024:20:29:09 +0000] "GET /api/v1/stock?symbol=OMCL HTTP/1.0" 502 157 "-" "-"
[07/Mar/2024:20:29:09 +0000] "GET /api/v1/stock?symbol=ACONW HTTP/1.0" 502 157 "-" "-"
[07/Mar/2024:20:29:09 +0000] "GET /api/v1/stock?symbol=OABI HTTP/1.0" 502 157 "-" "-"
[07/Mar/2024:20:29:09 +0000] "GET /api/v1/stock?symbol=OMCL HTTP/1.0" 502 157 "-" "-"
[07/Mar/2024:20:29:09 +0000] "GET /api/v1/stock?symbol=APPL HTTP/1.0" 502 157 "-" "-"
[07/Mar/2024:20:29:09 +0000] "GET /api/v1/stock?symbol=ADNWW HTTP/1.0" 502 157 "-" "-"
[07/Mar/2024:20:29:09 +0000] "GET /api/v1/stock?symbol=OCGIO HTTP/1.0" 502 157 "-" "-"
```

Но есть один нюанс...

```
20 @Throws(BeansException::class) new *
21 @↑ override fun postProcessBeforeInitialization(bean: Any, beanName: String): Any? {
22     if (beanName == "expensiveLoader") {
23         runBlocking { this: CoroutineScope
24             logger.info { "Looks like we have a special scanner here >.<" }
25
26             for (i in 0 ≤ .. ≤ 60) {
27                 logger.info { "scan: $i" }
28                 delay(1.seconds)
29             }

```



Что важно помнить?

Не забывайте про health checks и правильной их настройкой.
Следите за пробамми и за временем их опроса.



Результаты

Один сервис - 2.7 мс 99%

Через NGINX – 15 мс 99%



Итоги!

- написали простые тесты на танке
- протестировали GET и POST
- сделали патроны для танка
- изучили конфиг и настройки теста
- проинспектировали сервис
- ускорили тривиальный поиск
- добавили балансировку ценой скорости



Call to action!

Тестировать нагрузкой достаточно просто и если у вас нет большой команды, вы стартап или просто хотите сами погрузиться в дебри тестирования - делайте это!



TODO

Нормирование нагрузки с помощью нагрузочного тестирования, но это "другая история"



Спасибо и желаю удачи!

