# Лекция 6

Node.js

Node.js как платформа

# Что такое Node.js?

Node.js - среда выполнения javascript основанная на браузерном движке Google V8.



# На что Node.js способен

- веб-приложения (бэкенд);
- десктопные приложения (<u>Electron.js</u>);
- микроконтроллеры (Espurino), <u>ссылка</u>;
- различные консольные утилиты;
- машинное обучение (TensorFlow.js), ссылка;
- бла-бла-бла...

# На что Node.js способен

- веб-приложения (бэкенд);
- десктопные приложения (<u>Electron.js</u>);
- микроконтроллеры (Espurino), <u>ссылка</u>;
- различные консольные утилиты;
- машинное обучение (TensorFlow.js), <u>ссылка</u>;
- бла-бла-бла...



Преимущества Node.js

# Скорость

Node.js - это платформа для разработки очень быстрого сервера. По производительности спокойно может сравниться с Java, Kotlin и Golang.

При этом все вышеперечисленные языки имею вложенные механизмы параллелизма (корутины, потоки и тд), а Node.js однопоточный.

Как так получается?

# **Асинхронное API**

Движок V8 из коробки поддерживает исключительно асинхронное API, т.к. изначально был предназначен для работы в браузере на клиенте. Это значит, что ни одна из read/write операций не блокирует основной поток исполнения (спасибо Event Loop).

# Простота

Поднять простой веб-сервер на Node.js - это 15 строк кода (с натяжкой).

При этом, т.к. мы пишем на javascript, мы можем использовать различные парадигмы для написания кода.

## Библиотеки

У Node.js огромное комьюнити, и с 2009 года существуют библиотеки для всего, что может теоретически понадобиться для серверной (и не только) разработки.

Количество javascript библиотек достигает такого огромного количества, что это уже даже достигло какого-то абсурдного уровня.





2.0.0 • Public • Published 7 years ago









2 Versions

### is-thirteen



Check if a number is equal to 13.

### Installation

npm --save i is-thirteen

### Usage

var isThirteen = require('is-thirteen');

### Install

> npm i is-thirteen

### Repository

• github.com/jezen/is-thirteen

### Homepage

 ${\cal S}$  github.com/jezen/is-thirteen#readme

Version

36

License

2.0.0

WTFPL

# **JavaScript**

Тут не нужно объяснений

Недостатки Node.js



16:54

У тебя было яблоко, тебе дали ещё одно

Сколько у тебя яблок?



16:55 Одно яблоко

Потому что яблоко + одно = одно яблоко



16:55

Твой родной язык это джаваскрипт что ли о

# JavaScript

## RunTime-ошибки

Это когда код запустился, но упал во время исполнения.

Такая возможность есть из-за того, что javascript интерпретируемый язык

программирования.

```
1 const a = undefined;
2 console.log('шалом ');
3
4 a.func(); // RunTime Exception
5
6 console.log('братья');
7
```

# Отсутствие типизации

Javascript - нестрого типизированный язык.

```
1 true + false
 2 12 / "6"
 3 \text{ "number"} + 15 + 3
 4 15 + 3 + "number"
 5 [1] > null
 6 "foo" + + "bar"
 7 'true' == true
 8 false == 'false'
 9 null == ''
10 !!"false" == !!"true"
11 ['x'] == 'x'
12 [] + null + 1
13 0 || "0" && {}
14 [1,2,3] == [1,2,3]
15 {}+[]+{}+[1]
16 !+[]+[]+![]
17 new Date(0) - 0
18 new Date(0) + 0
19
```

# Риск наговнокодить

Риск написать плохой код на JavaScript настолько высок, что сравнить его можно, наверно, только с Python. Обилие поддерживаемых парадигм, нестрогая типизация, сложная асинхронность, лексическое окружение, замыкания.

```
async registration(Form) {
    console.log('ajax post');
    return await ajax.post(backend.upload, new FormData(Form), true)
        .then(({status, responseObject}) => {
            let photo_name;
            if (status === 200 ) {
                photo name = new Promise((resolve, reject) => {
                    resolve(JSON.stringify(responseObject));
                    this.linkImage.push(responseObject.replaceAll('"', ''));
               });
                console.log(photo_name);
                return ajax.post(backend.signup, this.Json());
            }
            if (status === 400) {
                throw new Error('Слишком большой размер фото, пожалуйста, загрузите фото меньшего размера');
            }
            if (status === 403) {
                throw new Error('Пожалуйста, загрузите фото с вашим лицом');
            }
            if (status === 500) {
                throw new Error('Неизвестная ошибка, пожалуйста, попробуйте позже');
       });
```

## Большая математика

До недавнего времени в javascript не поддерживались числа больше 9007199254740992. Недавно появился bigint и эта проблема частично решена.

```
1 const bigNumber = 9007199254740992;
2
3 console.log(bigNumber); // Infinity
4
```

# А также в главных ролях

- отсутствие многопоточки;
- медленные вычисления;
- плохое ООП, несмотря на его наличие;
- скудные коллекции (по сравнению с java, но великолепные по-сравнению с Go)
- <подставьте сюда свою причину>

Особенности и использование

# Как использовать Node.js

- 1. Создаем файл <название>.js
- 2. Пишем туда код на javascript
- 3. Запускаем командой node <название>.js
- 4. Profit!

# Стандартная библиотека

- events встроенный EventEmitter
- fs работа с файловой системой
- os работа с операционной системой
- http основной модуль для создания http-серверов
- net низкоуровневое сетевое соединение
- раth для работы с путями в директориях
- и еще много-много, почитать можно здесь

# CommonJS модули

```
1 // file.js
 3 \text{ let count} = 0
 4 const increase = () => ++count
 5 const reset = () => {
    count = 0
      console.log('Счетчик сброшен.')
 8 }
10 exports.increase = increase
11 exports.reset = reset
12
13 // или (эквивалентно)
14 module.exports = {
15
      increase,
16
      reset
17 }
18
```

```
1 // используем CommonJS модуль
 2 const {
      increase,
 4 reset
 5 } = require('./file')
 6
 7 increase()
8 reset()
10 // или
11 const file = require('./file')
12 file.increase()
13 file.reset()
14
```

# global и globalThis

global - аналог window в среде Node.js.

Если нет уверенности, где будет исполняться ваш код, или использоваться написанная вами библиотека - используйте globalThis.

# Так как его используют? Этот ваш Node.js

Как уже повторяли для серверной разработки. Например:

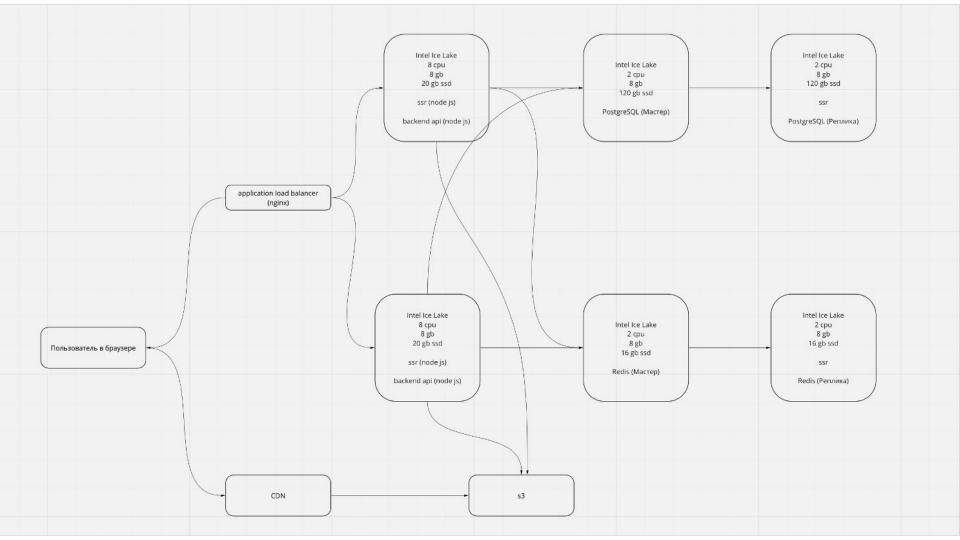
- SSR серверный рендеринг для SPA
- BFF бэкенд-прокси для агрегации данных
- Full-Backend API полноценный бэкенд
- различные микросервисы со специфической логикой

Поглядим примерчики

# А что там все таки по скорости? Давайте посмотрим пример

# **RPS**

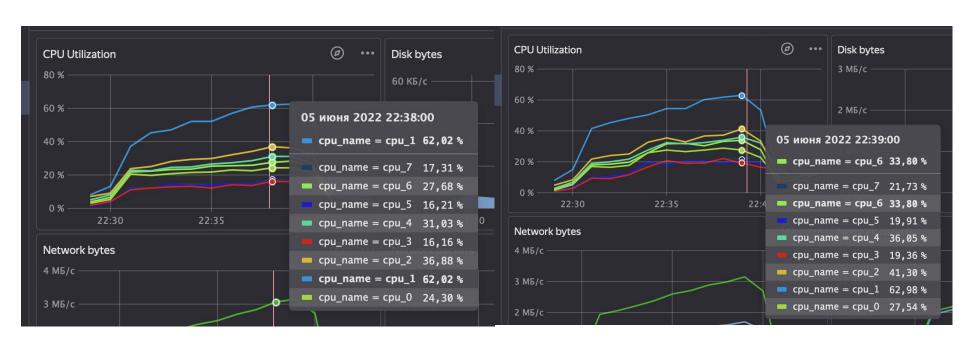
RPS - это метрика, которая определяет сколько запросов в секунду выдерживает сервис. Необходим для оценки нагрузки от пользователей и зачастую используется в расчетах.



# Высоконагруженность 200 rps

```
Data delay: 3s, RPS: 202
                                                                             . Duration: 0:10:00
                                                                                                       ETA: 0:00:00
Percentiles (all/last 1m/last), ms: . HTTP codes:
                                                                             . Hosts: ef31i12ouueo70d8ep16 => 51.250
100.0% < 5,060.0 1,085.0 149.2
                                   . 89,498 +202 100.00% : 200 OK
                                                                             Ammo:
99.5% < 489.0 99.0 142.0
                                                                             . Count: 90000
           262.0
                     87.0 122.3
99.0% <
                                    . Net codes:
                                                                              Load: line(100, 200, 10m)
95.0% <
                     66.0
                          68.7
         71.0
                                    . 89,498 +202 100.00% : 0 Success
90.0% <
            61.0
                     60.0
                           61.2
                                                                             . Active instances: 14
                                    . Average Sizes (all/last), bytes:
85.0% <
            56.0
                     56.0
                           59.2
                                                                             . Planned requests: 200.0 for 0:00:00
80.0% <
            54.0
                     54.0
                           56.2
                                                   56.0 /
                                    . Request:
                                                              56.0
                                                                             . Actual responses: 202
75.0% <
            52.0
                     52.0
                           54.0
                                    . Response: 23,462.9 / 23,462.7
                                                                                      Accuracy: 0.00%
70.0% <
            50.0
                     51.0
                           53.0
                                                                                      Time lag: 0:00:00
60.0% <
            48.0
                     48.0
                           50.0
                                    . Average Times (all/last), ms:
50.0% <
            46.0
                     46.0
                           47.3
                                    . Overall: 53.65 / 50.06
40.0% <
                           45.0
            44.0
                     44.0
                                    . Connect: 3.73 / 1.20
30.0% <
            42.0
                     42.0
                           44.0
                                        Send: 0.01 / 0.01
20.0% <
            41.0
                     41.0
                           42.0
                                    . Latency: 43.91 / 42.82
10.0% <
            39.0
                     38.0
                           39.0
                                    . Receive: 6.00 / 6.02
Cumulative Cases Info:
                     % last net_e http_e ava ms last ms
   name count
OVERALL: 89,498 100.00% +202
                                          53.7
                                                  50.1
```

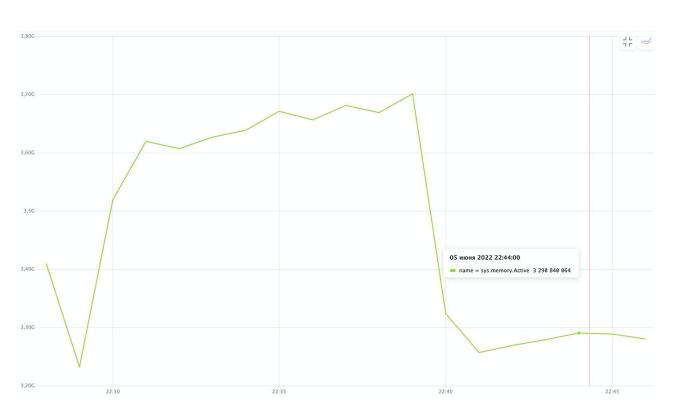
# Высоконагруженность 200 rps



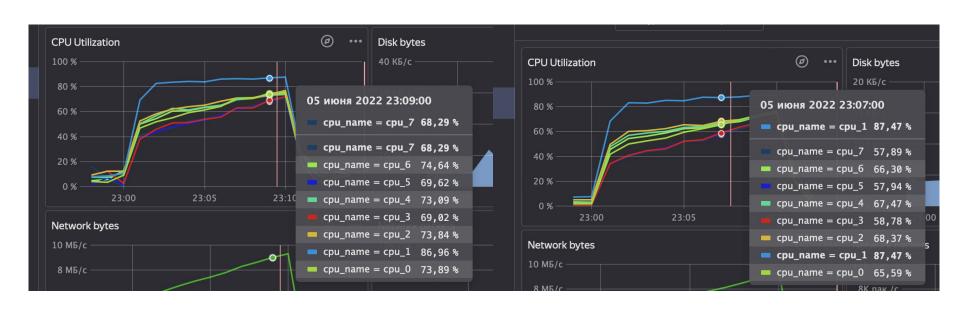
# Высоконагруженность 200 rps

<pre>Process List —</pre>		
[8] svarog-api	Mem: 123 MB	CPU: 0 % onlin
[ 9] svarog-api	Mem: 123 MB	CPU: 0 % onlin
[10] svarog-api	Mem: 124 MB	CPU: 6 % onlin
[11] svarog-api	Mem: 128 MB	CPU: 4 % onlin
[12] svarog-api	Mem: 123 MB	CPU: 4 % onlin
[13] svarog-api	Mem: 130 MB	CPU: 2 % onlin
[14] svarog-api	Mem: 125 MB	CPU: 2 % onlin
[15] svarog-api	Mem: 128 MB	CPU: 4 % onlin
[26] svarog-ui	Mem: 193 MB	CPU: 15 % online
[27] svarog-ui	Mem: 198 MB	CPU: 2 % online
[28] svarog-ui	Mem: 212 MB	CPU: 37 % online
[29] svarog-ui	Mem: 195 MB	CPU: 15 % online
[30] svarog-ui	Mem: 195 MB	CPU: 13 % online
[31] svarog-ui	Mem: 187 MB	CPU: 19 % online
[32] svarog-ui	Mem: 203 MB	CPU: 4 % online
[33] svarog-ui	Mem: 202 MB	CPU: 8 % online

```
Mem: 123 MB
                                              onlin
 8] svarog-api
                                   CPU:
 9] svarog-api
                                              onlin
                    Mem: 123 MB
                                   CPU:
[10] svarog-api
                                   CPU:
                                              onlin
                    Mem: 125 MB
                                   CPU:
[11] svarog-api
                    Mem: 128 MB
                                              onlin
[12] svarog-api
                    Mem: 123 MB
                                   CPU:
                                              onlin
[13] svarog-api
                    Mem: 131 MB
                                   CPU:
                                              onlin
[14] svarog-api
                   Mem: 125 MB
                                   CPU:
                                              onlin
[15] svarog-api
                    Mem: 128 MB
                                   CPU:
                                         0 % onlin
[26] svarog-ui
                   Mem: 196 MB
                                  CPU: 14 % online
[27] svarog-ui
                   Mem: 198 MB
                                  CPU:
                                       2 % online
[28] svarog-ui
                   Mem: 224 MB
                                       41 % online
[29] svarog-ui
                   Mem: 199 MB
                                  CPU:
                                          % online
[30] svarog-ui
                   Mem: 195 MB
                                       12 % online
[31] svarog-ui
                   Mem: 187 MB
                                  CPU:
                                        2% online
[32] svarog-ui
                   Mem: 153 MB
                                        9 % online
[33] svarog-ui
                                  CPU: 22 % online
                   Mem: 201 MB
```

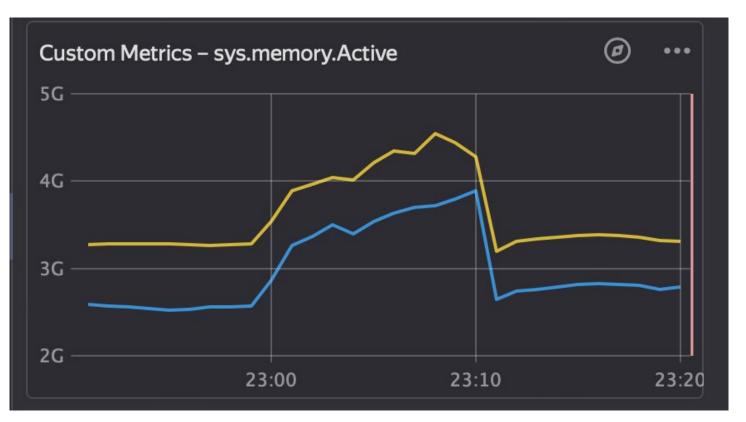


```
99.5%
Data delay: 3s, RPS: 512
                                                                            . Duration: 0:10:01
                                                                                                      ETA: 0:00:02
Percentiles (all/last 1m/last), ms: . HTTP codes:
                                                                            . Hosts: ef31i12ouueo70d8epl6 => 51.250
100.0% < 5,780.0 5,210.0
                          202.9 . 238,634 +512 99.90% : 200 OK
                                                                               Ammo:
99.5% < 1.100.0 1.535.0 199.0 .
                                         246
                                             +0 0.10%: 500 Internal Se... . Count: 240000
                             194.4
                                                   0.00%: 503 Service Una... Load: line(300, 500, 10m)
99.0% <
           775.0 1,215.0
 95.0% <
           1//.0
                    555.0
                            162.9 .
90.0% <
           131.0
                    285.0
                            150.0 . Net codes:
                                                                            . Active instances: 77
85.0% <
           114.0
                    173.0
                                   . 238,883 +512 100.00% : 0 Success
                                                                            . Planned requests: 499.0 for 0:00:00
                            133.9
80.0% <
           104.0
                   147.0
                                                                            . Actual responses: 512
75.0% <
            97.0
                    134.0
                            129.1 . Average Sizes (all/last), bytes:
                                                                                     Accuracy: 0.00%
                                                                                     Time lag: 0:00:00
70.0% <
            91.0
                    124.0
                            123.0
                                   . Request:
                                                  56.0 /
                                                             56.0
60.0% <
            82.0
                    110.0
                            115.0 . Response: 27,262.2 / 27,279.4
50.0% <
            74.0
                    100.0
                            106.3
40.0% <
            68.0
                    91.0
                             99.0 . Average Times (all/last), ms:
30.0% <
            63.0
                    83.0
                             91.0
                                  . Overall: 100.78 / 109.00
20.0% <
            57.0
                                               4.60 / 1.39
                    75.0
                             81.0
                                   . Connect:
10.0% <
            51.0
                     66.0
                             72.0
                                        Send:
                                               0.01 /
                                                        0.01
                                              87.52 / 97.66
                                    . Latency:
                                    . Receive:
                                               8.65 /
                                                        9.94
Cumulative Cases Info:
                      % last net_e http_e avg ms last ms
   name
          count
OVERALL: 238,883 100.00% +512
                                     249 100.8
                                                  109.0
```

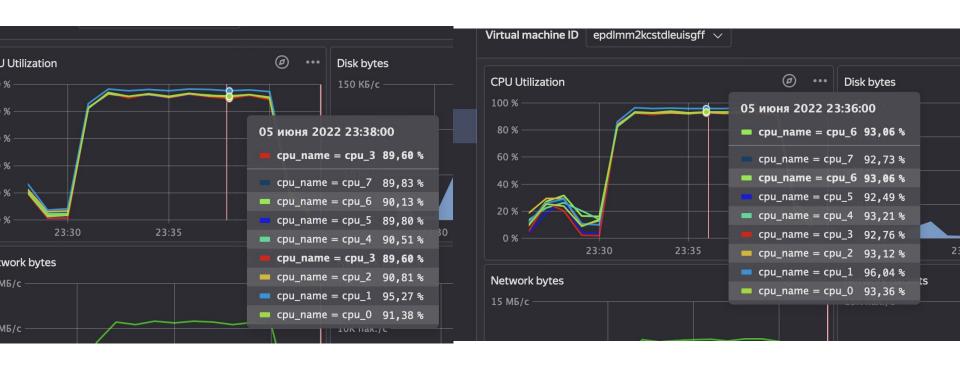


<pre>Process List —</pre>		
[ 8] svarog-api	Mem: <b>126</b> MB	CPU: 28 % onlin
[ 9] svarog-api	Mem: 127 MB	CPU: 19 % onlin
[10] svarog-api	Mem: 128 MB	CPU: 8 % onlin
[11] svarog-api	Mem: 130 MB	CPU: 14 % onlin
[12] svarog-api	Mem: 101 MB	CPU: 26 % onlin
[13] svarog-api	Mem: 130 MB	CPU: 14 % onlin
[14] svarog-api	Mem: 126 MB	CPU: 21 % onlin
[15] svarog-api	Mem: 129 MB	CPU: 11 % onlin
[26] svarog-ui	Mem: 373 MB	CPU: 60 % online
[27] svarog-ui	Mem: 268 MB	CPU: 85 % online
[28] svarog-ui	Mem: 269 MB	CPU: 55 % online
[29] svarog-ui	Mem: 291 MB	CPU: 68 % online
[30] svarog-ui	Mem: 265 MB	CPU: 58 % online
[31] svarog-ui	Mem: 248 MB	CPU: 71 % online
[32] svarog-ui	Mem: 231 MB	CPU: 46 % online
[33] svarog-ui	Mem: 364 MB	CPU: 40 % online

Process List		
[ 8] svarog-api	Mem: <b>127</b> MB	CPU: 18 % onlin
[ 9] svarog-api	Mem: 127 MB	CPU: 12 % onlin
[10] svarog-api	Mem: 123 MB	CPU: 4 % onlin
[11] svarog-api	Mem: 137 MB	CPU: 16 % onlin
[12] svarog-api	Mem: 90 MB	CPU: 24 % onlin
[13] svarog-api	Mem: 131 MB	CPU: 10 % onlin
[14] svarog-api	Mem: 130 MB	CPU: 25 % onlin
[15] svarog-api	Mem: 130 MB	CPU: 1 % onlin
[26] svarog-ui	Mem: 308 MB	CPU: 21 % online
[27] svarog-ui	Mem: 279 MB	CPU: 39 % online
[28] svarog-ui	Mem: 275 MB	CPU: 40 % online
[29] svarog-ui	Mem: 327 MB	CPU: 62 % online
[30] svarog-ui	Mem: 237 MB	CPU: 25 % online
[31] svarog-ui	Mem: 267 MB	CPU: 22 % online
[32] svarog-ui	Mem: 246 MB	CPU: 47 % online
[33] svarog-ui	Mem: 368 MB	CPU: 54 % online

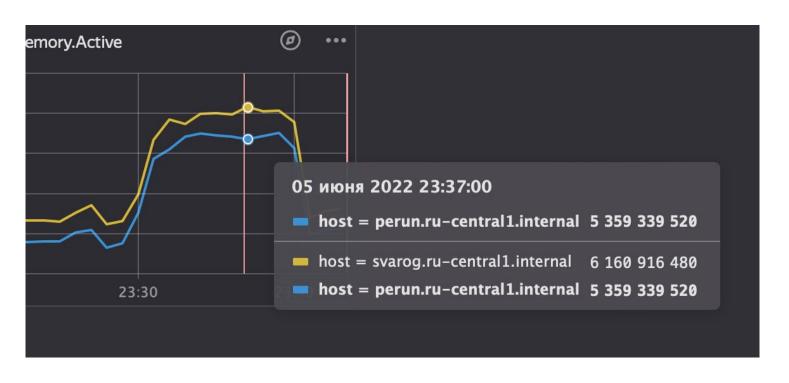


```
Data delay: 4s, RPS: 791
                                                                             Duration: 0:00:10
                                                                                                     ETA: 0:09:50
Percentiles (all/last 1m/last), ms: . HTTP codes:
                                                                           . Hosts: ef31i12ouueo70d8epl6 => 51.250
100.0\% < 4,110.0 4,110.0 4,100.7 . 2.537 +460
                                                                              Ammo:
 QQ 50/ 2 150 0 2 150 0 2 QQ0 0
                                                                             Count: 420000
 99.0% < 3,070.0 3,070.0 3,756.6
                                                                              Load: line(600, 800, 10m)
 95.0\% < 2,620.0 2,620.0 2,808.1 .
 90.0% < 2,255.0 2,255.0 2,628.7 . Net codes:
                                                                           . Active instances: 692
 85.0% < 1,980.0 1,980.0 2,441.9 . 3,393 +791 100.00% : 0 Success
                                                                           . Planned requests: 602.0 for20:27:12 [WARNING] Autostop
20:27:12 [WARNING] Autostop criterion requested test stop: 5xx codes count higher than 10.0% for 5s, since 1654460825
0:00:00
 80.0% < 1,810.0 1,810.0 2,027.4 .
                                                                           . Actual responses: 791
 75.0% < 1.645.0 1,645.0 1,890.4 . Average Sizes (all/last), bytes:
                                                                                     Accuracy: 0.00%
 70.0% < 1,505.0 1,505.0 1,785.0 . Request:
                                                                                     Time laa: 0:00:00
                                                   58.9 /
                                                             58.6
 60.0% < 1,250.0 1,250.0 1,220.0 . Response: 19,920.8 / 15,752.2
 50.0% < 1,010.0 1,010.0
                            733.2 .
                                                                           . Autostop:
 40.0% <
           730.0
                  730.0
                            525.0 . Average Times (all/last), ms:
                                                                               HTTP 5xx>10.0% for 4/5s
 30.0% <
           418.0
                  418.0
                           2.3 . Overall: 1,046.88 / 1,043.15
 20.0% <
                              0.6 . Connect:
            11.0
                   11.0
                                                 2.62 /
                                                            0.00
 10.0% <
             0.5
                                        Send:
                                                 0.02 /
                                                            0.02
                      0.5
                              0.4 .
                                   . Latency: 1,019.59 / 1,027.45
                                   . Receive:
                                                24.66 /
                                                           15.68
Cumulative Cases Info:
                    % last net_e http_e avg ms last ms
    name count
OVERALL: 3,393 100.00% +791
```

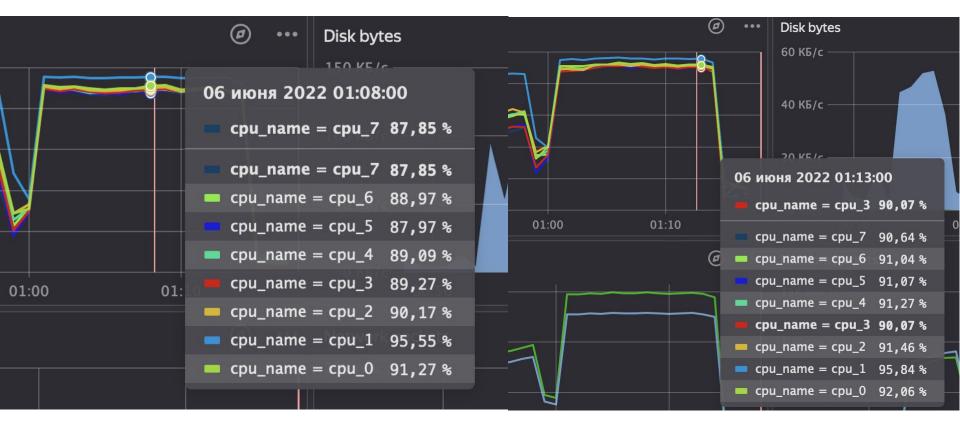


[8] svarog-api	Mem: <b>224</b> MB	CPU: <b>15</b> % onlin
[ 9] svarog-api	Mem: 201 MB	CPU: 10 % onlin
[10] svarog-api	Mem: 152 MB	CPU: 19 % onlin
[11] svarog-api	Mem: 231 MB	CPU: 11 % onlin
[12] svarog-api	Mem: 163 MB	CPU: 24 % onlin
[13] svarog-api	Mem: 159 MB	CPU: 20 % onlin
[14] svarog-api	Mem: 153 MB	CPU: 24 % onlin
[15] svarog-api	Mem: 145 MB	CPU: 14 % onlin
[26] svarog-ui	Mem: 481 MB	CPU: 64 % online
[27] svarog-ui	Mem: 314 MB	CPU: 69 % online
[28] svarog-ui	Mem: 462 MB	CPU: 65 % online
[29] svarog-ui	Mem: 393 MB	CPU: 68 % online
[30] svarog-ui	Mem: 467 MB	CPU: 65 % online
[31] svarog-ui	Mem: 480 MB	CPU: 67 % online
[32] svarog-ui	Mem: 447 MB	CPU: 69 % online
[33] svarog-ui	Mem: 468 MB	CPU: 65 % online

```
Mem: 136 MB
                                  CPU: 9 % onlin
 87 svarog-api
                                   CPU: 11 % onlin
[ 9] svaroq-api
                   Mem: 214 MB
                   Mem: 216 MB
                                  CPU: 11 % onlin
[10] svarog-api
[11] svarog-api
                   Mem: 205 MB
                                  CPU: 5 % onlin
[12] svarog-api
                   Mem: 159 MB
                                  CPU: 27 %
[13] svaroq-api
                   Mem: 136 MB
                                  CPU: 15 % onlin
[14] svarog-api
                                  CPU: 23 % onlin
                   Mem: 159 MB
[15] svarog-api
                                  CPU: 23 % onlin
                   Mem: 145 MB
                                  CPU: 64 % online
[26] svarog-ui
                  Mem: 455 MB
[27] svarog-ui
                  Mem: 411 MB
                                  CPU: 86 % online
[28] svaroq-ui
                  Mem: 460 MB
                                  CPU: 60 % online
[29] svaroq-ui
                  Mem: 375 MB
                                 CPU: 70 % online
                                  CPU: 68 % online
[30] svarog-ui
                  Mem: 483 MB
                  Mem: 482 MB
[31] svarog-ui
                                  CPU: 70 % online
[32] svarog-ui
                  Mem: 466 MB
                                 CPU: 39 % online
[33] svarog-ui
                  Mem: 468 MB
                                  CPU: 70 % online
```

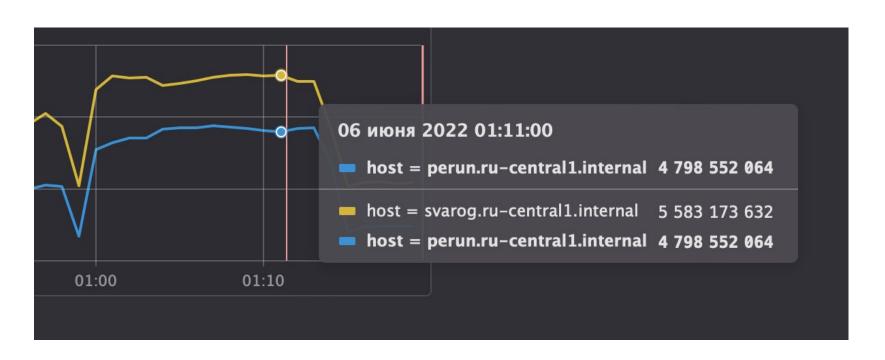


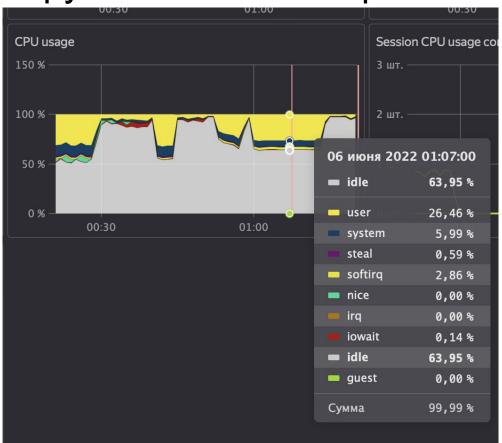
```
99.6%
Data delay: 3s, RPS: 5,216
                                                                            . Duration: 0:14:17
                                                                                                      ETA: 0:00:02
Percentiles (all/last 1m/last), ms:
                                   . HTTP codes:
                                                                            . Hosts: ef31i12ouueo70d8epl6 => 51.250
100.0% < 1,550.0 1,140.0 935.4
                                    . 4,486,190 +5,216 100.00% : 200 OK
                                                                            . Ammo:
99 5% - 860 0 990 0 910 0
                                           144
                                                        0.00%: 503 Servic... . Count: 4500000
 99.0% <
           810.0
                    915.0 897.2
                                                                            . Load: line(6000, 9000, 10m)
                                   . Net codes:
                    כ.בכס ש.שכט
 უე.ს/ს <___
           ש.כשכ
90.0% <
           347.0
                    337.0 360.8
                                    . 4,486,334 +5,216 100.00% : 0 Success
                                                                            . Active instances: 1000
85.0% <
           300.0
                    288.0 309.0
                                                                            . Planned requests: 0 for 0:00:00
80.0% <
                    258.0 254.9
                                   . Average Sizes (all/last), bytes:
           270.0
                                                                            . Actual responses: 5216
75.0% <
           248.0
                    235.0 231.7
                                   . Request: 71.5 / 71.5
                                                                                     Accuracy: 0.00%
70.0% <
                    218.0 220.0
                                   . Response: 3,644.1 / 3,649.2
           229.0
                                                                                     Time lag: 0:00:00
60.0% <
                    186.0 190.0
           194.0
50.0% <
                    150.0 162.5
                                   . Average Times (all/last), ms:
           160.0
40.0% <
                    109.0 105.0
           125.0
                                   . Overall: 190.38 / 194.57
30.0% <
            93.0
                    80.0
                           80.0
                                   . Connect:
                                               0.01 / 0.00
20.0% <
            66.0
                     58.0
                           60.0
                                        Send:
                                               0.01 / 0.01
10.0% <
            44.0
                     41.0 42.0
                                   . Latency: 190.19 / 194.40
                                               0.17 / 0.17
                                    . Receive:
Cumulative Cases Info:
                          last net_e http_e ava ms last ms
            count
   name
OVERALL: 4,486,334 100.00% +5,216
                                         144 190.4 194.6
```



<pre>- Process List</pre>				
[ 8] svarog-api	Mem: 257 MB	CPU:	29 %	onlin
[ 9] svarog-api	Mem: 332 MB	CPU:	82 %	onlin
[10] svarog-api	Mem: 268 MB	CPU:	81 %	onlin
[11] svarog-api	Mem: 280 MB	CPU:	90 %	onlin
[12] svarog-api	Mem: 280 MB	CPU:	97 %	onlin
[13] svarog-api	Mem: 281 MB	CPU:	81 %	onlin
[14] svarog-api	Mem: 315 MB	CPU:	89 %	onlin
[15] svarog-api	Mem: 312 MB	CPU:	86 %	onlin
[26] svarog-ui	Mem: 241 MB	CPU:	2 %	online
[27] svarog-ui	Mem: 245 MB	CPU:	4 %	online
[28] svarog-ui	Mem: 228 MB	CPU:	2 %	online
[29] svarog-ui	Mem: 246 MB	CPU:	1 %	online
[30] svarog-ui	Mem: 246 MB	CPU:	4 %	online
[31] svarog-ui	Mem: 248 MB	CPU:	2 %	online
[32] svarog-ui	Mem: 253 MB	CPU:	2 %	online
[33] svarog-ui	Mem: 251 MB	CPU:	0 %	online

```
Mem: 273 MB
                                   CPU: 21 %
                                               onlin
    svarog-api
   svarog-api
                    Mem: 334 MB
                                   CPU: 83 %
                                               onlin
[10] svarog-api
                    Mem: 257 MB
                                   CPU: 75 %
                                              onlin
[11] svarog-api
                    Mem: 277 MB
                                   CPU: 92 %
                                              onlin
[12] svarog-api
                    Mem: 299 MB
                                   CPU: 87 %
                                              onlin
                                   CPU: 83 %
[13] svarog-api
                    Mem: 275 MB
                                              onlin
[14] svarog-api
                    Mem: 314 MB
                                   CPU: 106 % onli
[15] svarog-api
                    Mem: 313 MB
                                   CPU: 90 % onlin
[26] svarog-ui
                   Mem: 242 MB
                                              online
                                   CPU:
                                        3 %
[27] svaroq-ui
                   Mem: 245 MB
                                  CPU:
                                         2 %
                                              online
[28] svarog-ui
                   Mem: 234 MB
                                  CPU:
                                         0 %
                                              online
[29] svarog-ui
                   Mem: 246 MB
                                  CPU:
                                              online
[30] svarog-ui
                   Mem: 246 MB
                                  CPU:
                                         5 %
                                              online
[31] svarog-ui
                   Mem: 248 MB
                                  CPU:
                                         2 %
                                              online
[32] svarog-ui
                   Mem: 253 MB
                                  CPU:
                                         0 %
                                              online
[33] svarog-ui
                   Mem: 251 MB
                                  CPU:
                                              online
```





А можно ли не городить свои костыли?

Можно! Давайте посмотрим на Nest.js

#### Nest.js

Nest.js - это фреймворк для создания эффективных, расширяемых серверных приложений.



#### Какие преимущества дает Nest.js

- готовую архитектуру;
- готовые решения для middleware;
- удобные механизмы валидации;
- возможность grpc почти из коробки;
- dependency injection
- java-подобное приложение.

Посмотрим примеры