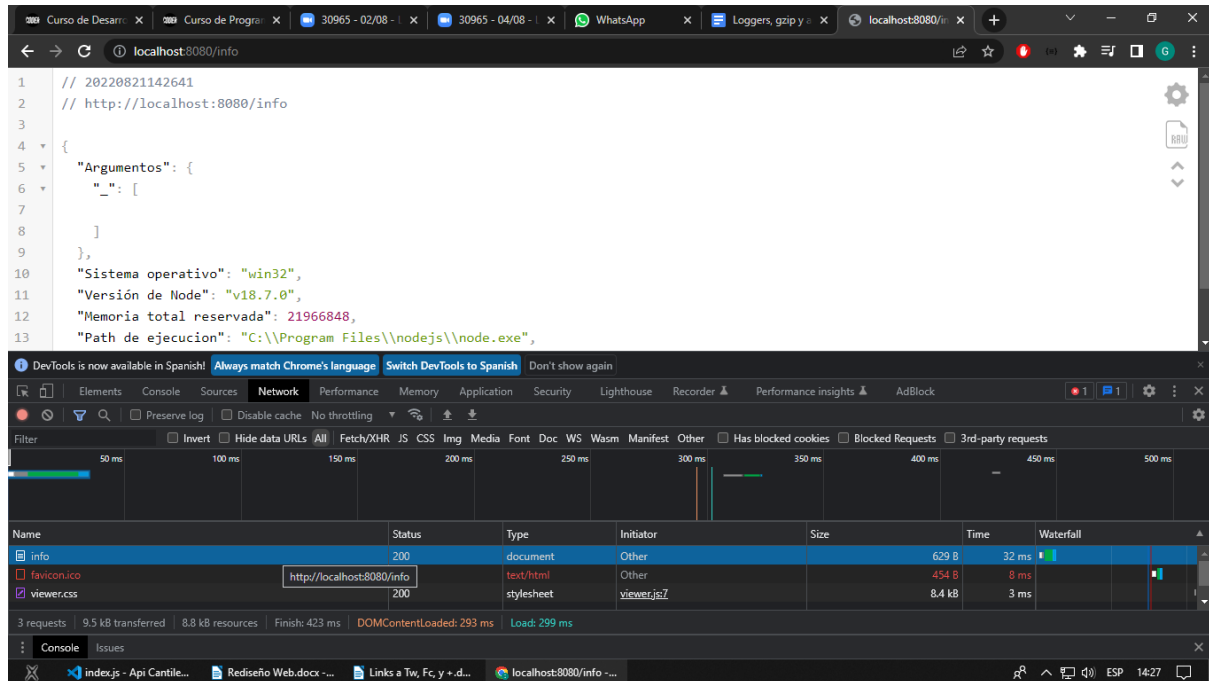


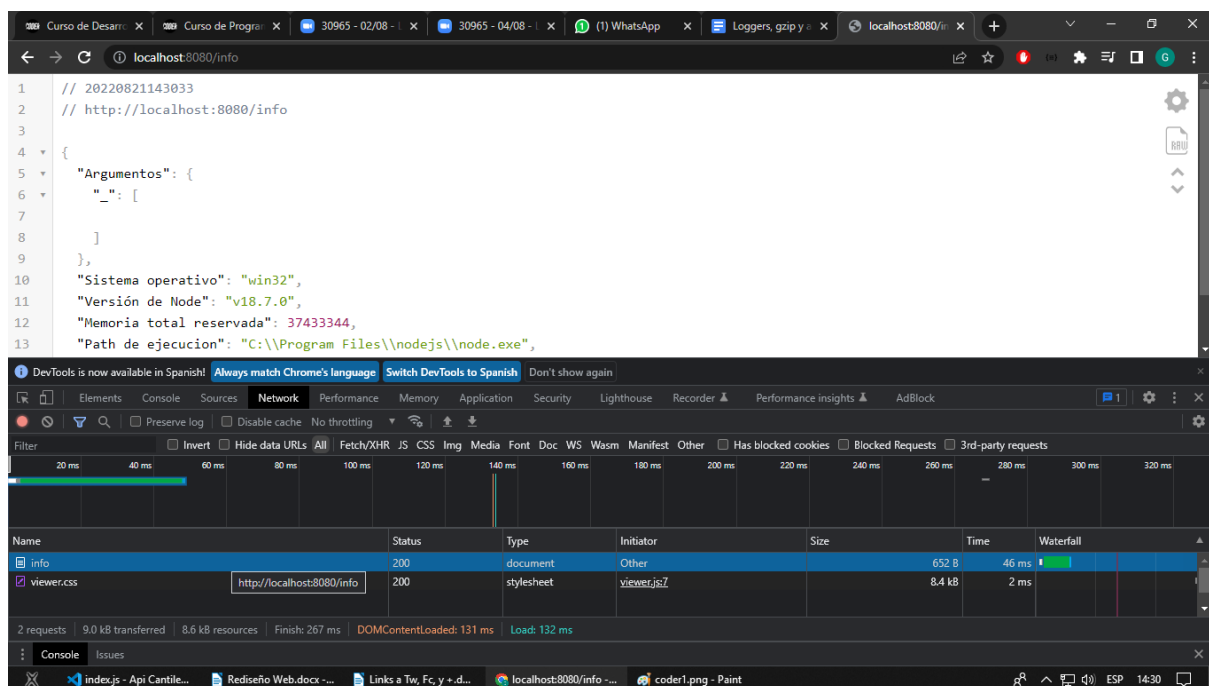
Loggers, gzip y análisis de performance

Verificación de compresión sobre “/info” =>

Sin Compresión =>



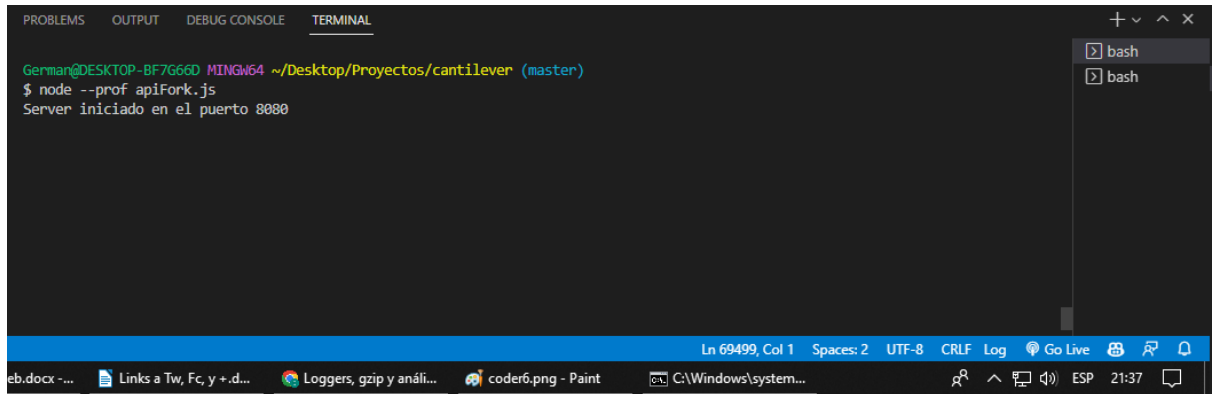
Con Compresión =>



Al tener poca información la respuesta, esta no se comprime.

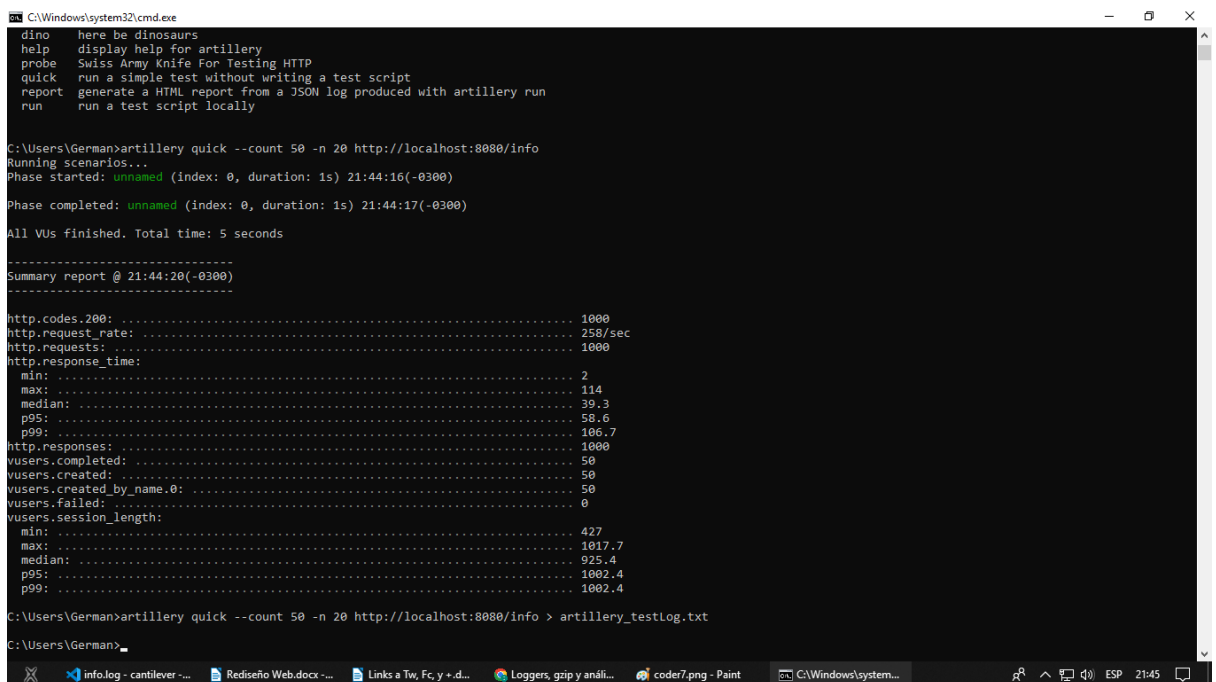
Analisis con -prof de node: =>

1. Iniciar la app: node -prof apiFork.js



```
German@DESKTOP-BF7G66D MINGW64 ~/Desktop/Proyectos/cantilever (master)
$ node -prof apiFork.js
Server iniciado en el puerto 8080
```

2. realizar el test con artillery (con console.log)



```
C:\Windows\system32\cmd.exe
dino here be dinosaurs
help display help for artillery
probe Swiss Army Knife For Testing HTTP
quick run a simple test without writing a test script
report generate a HTML report from a JSON log produced with artillery run
run run a test script locally

C:\Users\German>artillery quick --count 50 -n 20 http://localhost:8080/info
Running scenarios...
Phase started: unnamed (index: 0, duration: 1s) 21:44:16(-0300)
Phase completed: unnamed (index: 0, duration: 1s) 21:44:17(-0300)
All VUs finished. Total time: 5 seconds

Summary report @ 21:44:20(-0300)

http.codes.200: ..... 1000
http.request_rate: ..... 258/sec
http.requests: ..... 1000
http.response_time:
  min: ..... 2
  max: ..... 114
  median: ..... 39.3
  p95: ..... 58.6
  p99: ..... 106.7
http.responses: ..... 1000
vusers.completed: ..... 50
vusers.created: ..... 50
vusers.created_by_name.0: ..... 50
vusers.failed: ..... 0
vusers.session_length:
  min: ..... 427
  max: ..... 1017.7
  median: ..... 925.4
  p95: ..... 1002.4
  p99: ..... 1002.4

C:\Users\German>artillery quick --count 50 -n 20 http://localhost:8080/info > artillery_testLog.txt
C:\Users\German>
```

3. realizar test con artillery (sin console.log)

```
C:\Windows\system32\cmd.exe

C:\Users\German>artillery quick --count 50 -n 20 http://localhost:8080/info > artillery_testUnLog.txt

C:\Users\German>artillery quick --count 50 -n 20 http://localhost:8080/info
Running scenarios...
Phase started: unnamed (index: 0, duration: 1s) 21:49:12(-0300)
Phase completed: unnamed (index: 0, duration: 1s) 21:49:13(-0300)
All VUs finished. Total time: 5 seconds

-----
Summary report @ 21:49:16(-0300)
-----

http.codes.200: ..... 1000
http.request_rate: ..... 311/sec
http.requests: ..... 1000
http.response_time:
  min: ..... 1
  max: ..... 49
  median: ..... 25.8
  p95: ..... 44.3
  p99: ..... 47
http.responses: ..... 1000
vusers.completed: ..... 50
vusers.created: ..... 50
vusers.created_by_name.0: ..... 50
vusers.failed: ..... 0
vusers.session_length:
  min: ..... 179.7
  max: ..... 734.1
  median: ..... 608
  p95: ..... 727.9
  p99: ..... 727.9

C:\Users\German>
```

Ahora convertir los prof generados a txt

1. sin console.log =>

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

German@DESKTOP-BF7G66D MINGW64 ~/Desktop/Proyectos/cantilever (master)
$ node --prof-process logs\NoLogs.log > noConsoleLog.txt
(node:4148) ExperimentalWarning: VM Modules is an experimental feature. This feature could change at any time
(Use `node --trace-warnings ...` to show where the warning was created)

German@DESKTOP-BF7G66D MINGW64 ~/Desktop/Proyectos/cantilever (master)
$
```

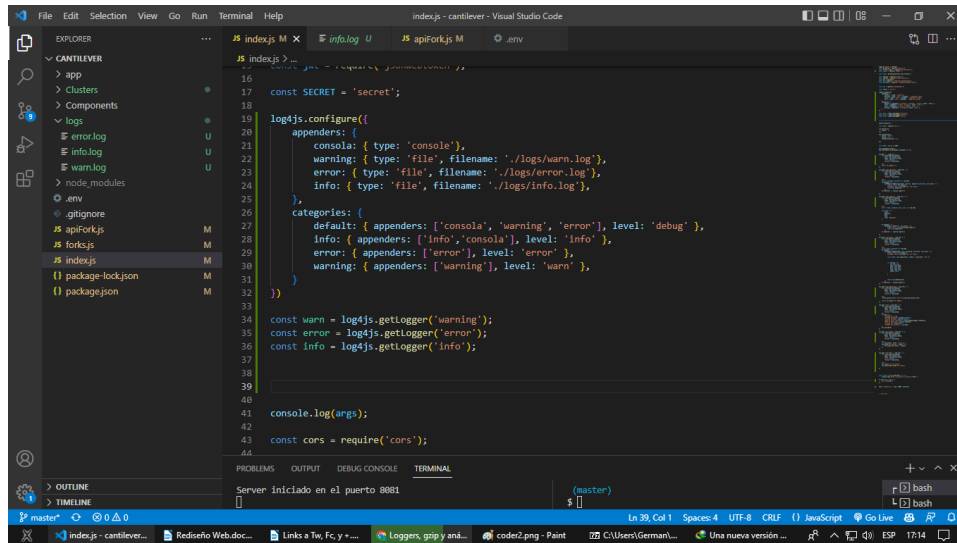
2. Con console.log

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

German@DESKTOP-BF7G66D MINGW64 ~/Desktop/Proyectos/cantilever (master)
$ node --prof-process logs\kconsole.log > ConsoleLog.txt
(node:2624) ExperimentalWarning: VM Modules is an experimental feature. This feature could change at any time
(Use `node --trace-warnings ...` to show where the warning was created)

German@DESKTOP-BF7G66D MINGW64 ~/Desktop/Proyectos/cantilever (master)
$
```

Integrar loggers de librería log4js=>

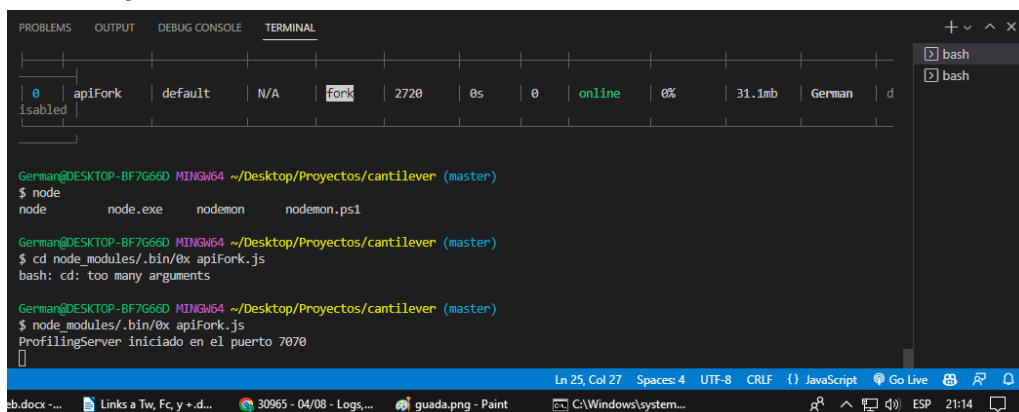


The screenshot shows a Visual Studio Code editor with a file named `index.js` open. The file contains a configuration for the `log4js` library. The configuration includes setting a secret, defining appenders for console, warning, error, and info logs, and setting up categories for default, info, error, and warning logs. The code also includes a `main` function that logs arguments and a `cors` module.

```
16 const SECRET = 'secret';
17
18
19 log4js.configure({
20   appenders: {
21     console: { type: 'console' },
22     warning: { type: 'file', filename: './logs/warn.log' },
23     error: { type: 'file', filename: './logs/error.log' },
24     info: { type: 'file', filename: './logs/info.log' },
25   },
26   categories: {
27     default: { appenders: ['console', 'warning', 'error'], level: 'debug' },
28     info: { appenders: ['info', 'console'], level: 'info' },
29     error: { appenders: ['error'], level: 'error' },
30     warning: { appenders: ['warning'], level: 'warn' },
31   }
32 })
33
34 const warn = log4js.getLogger('warning');
35 const error = log4js.getLogger('error');
36 const info = log4js.getLogger('info');
37
38
39
40
41 console.log(args);
42
43 const cors = require('cors');
```

Análisis de performance

1. usar 0x para iniciar el servidor en modo fork =>



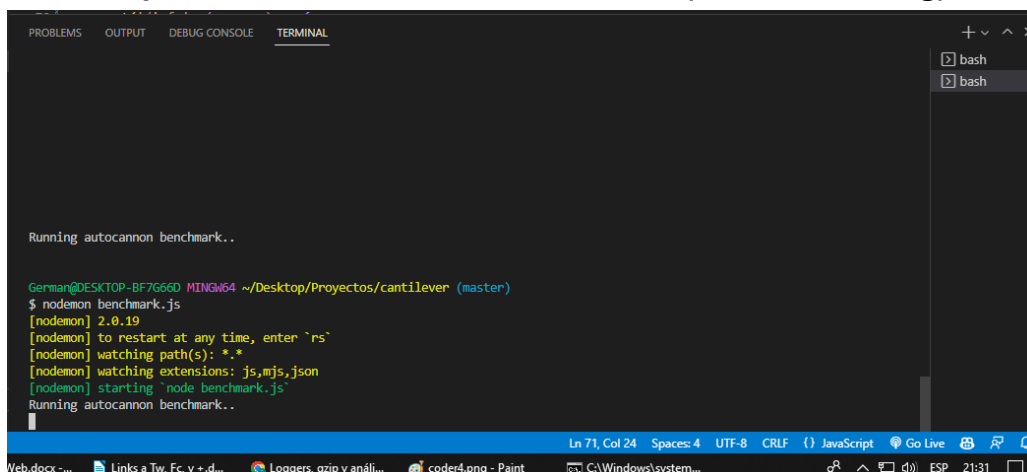
The screenshot shows a Visual Studio Code terminal window. The terminal displays the output of the `node` command, showing the process list. The `node` command is run, and the output shows the process list. The `node` command is run, and the output shows the process list. The `node` command is run, and the output shows the process list.

```
German@DESKTOP-BF7G66D MINGW64 ~/Desktop/Proyectos/cantilever (master)
$ node
node      node.exe      nodemon      nodemon.ps1

German@DESKTOP-BF7G66D MINGW64 ~/Desktop/Proyectos/cantilever (master)
$ cd node_modules/.bin/0x apiFork.js
bash: cd: too many arguments

German@DESKTOP-BF7G66D MINGW64 ~/Desktop/Proyectos/cantilever (master)
$ node_modules/.bin/0x apiFork.js
ProfilingServer iniciado en el puerto 7070
```

2. Iniciar la prueba de rendimiento con cannon (con console log) =>



The screenshot shows a Visual Studio Code terminal window. The terminal displays the output of the `node` command, showing the process list. The `node` command is run, and the output shows the process list. The `node` command is run, and the output shows the process list.

```
Running autocannon benchmark..

German@DESKTOP-BF7G66D MINGW64 ~/Desktop/Proyectos/cantilever (master)
$ nodemon benchmark.js
[nodemon] 2.0.19
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node benchmark.js`
Running autocannon benchmark..
```

3. Resultados de cannon =>

```
[nodemon] starting `node benchmark.js`
Running autocannon benchmark..
Running 30s test @ http://localhost:8080/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stddev	Max
Latency	16 ms	40 ms	82 ms	94 ms	42.72 ms	16.13 ms	172 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stddev	Min
Req/Sec	1177	1177	2431	2773	2313.14	371.31	1177
Bytes/Sec	630 kB	630 kB	1.3 MB	1.49 MB	1.24 MB	199 kB	630 kB

Req/Bytes counts sampled once per second.
of samples: 30

69k requests in 30.06s, 37.1 MB read
[nodemon] clean exit - waiting for changes before restart

Ln 71, Col 24 Spaces: 4 UTF-8 CRLF () JavaScript Go Live

4. Finalizar 0x con ctrl + c para generar el diagrama de flama =>

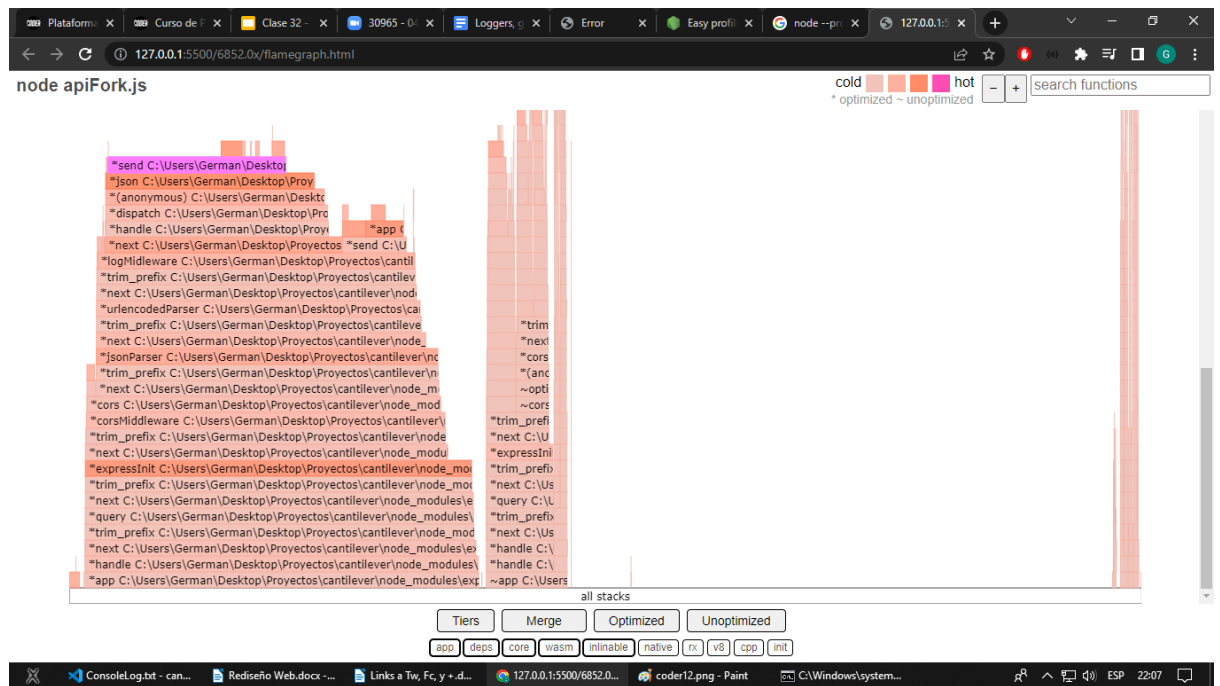
```
[2022-08-21T22:05:28.112] [INFO] info - { method: 'GET', url: '/info', body: {}, params: {}, query: {} }
[2022-08-21T22:05:28.112] [INFO] info - { method: 'GET', url: '/info', body: {}, params: {}, query: {} }
[2022-08-21T22:05:28.113] [INFO] info - { method: 'GET', url: '/info', body: {}, params: {}, query: {} }
[2022-08-21T22:05:28.113] [INFO] info - { method: 'GET', url: '/info', body: {}, params: {}, query: {} }
[2022-08-21T22:05:28.126] [INFO] info - { method: 'GET', url: '/info', body: {}, params: {}, query: {} }
[2022-08-21T22:05:28.127] [INFO] info - { method: 'GET', url: '/info', body: {}, params: {}, query: {} }
[2022-08-21T22:05:28.128] [INFO] info - { method: 'GET', url: '/info', body: {}, params: {}, query: {} }
[2022-08-21T22:05:28.128] [INFO] info - { method: 'GET', url: '/info', body: {}, params: {}, query: {} }
[2022-08-21T22:05:28.129] [INFO] info - { method: 'GET', url: '/info', body: {}, params: {}, query: {} }
[2022-08-21T22:05:28.130] [INFO] info - { method: 'GET', url: '/info', body: {}, params: {}, query: {} }
```

Flamegraph generated in
file://C:\Users\German\Desktop\Proyectos\cantilever\6852.0x\flamegraph.html

German@DESKTOP-BF7G66D MINGW64 ~/Desktop/Proyectos/cantilever (master)
\$

Ln 1, Col 1 Spaces: 2 UTF-8 LF Plain Text Go Live

5. diagrama de flama =>



Todo esto da como referencia que utilizar apps en modo fork es un impedimento de performance que puede ser solucionado implementando clusters para aprovechar mejor el rendimiento