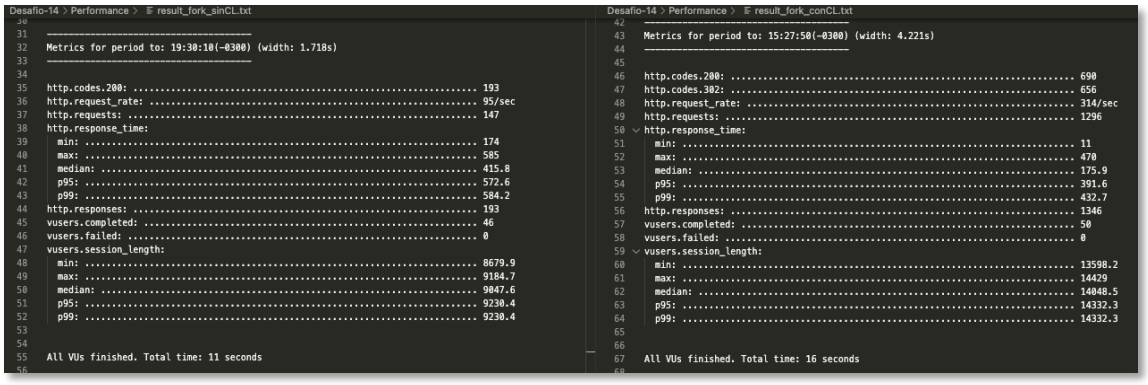


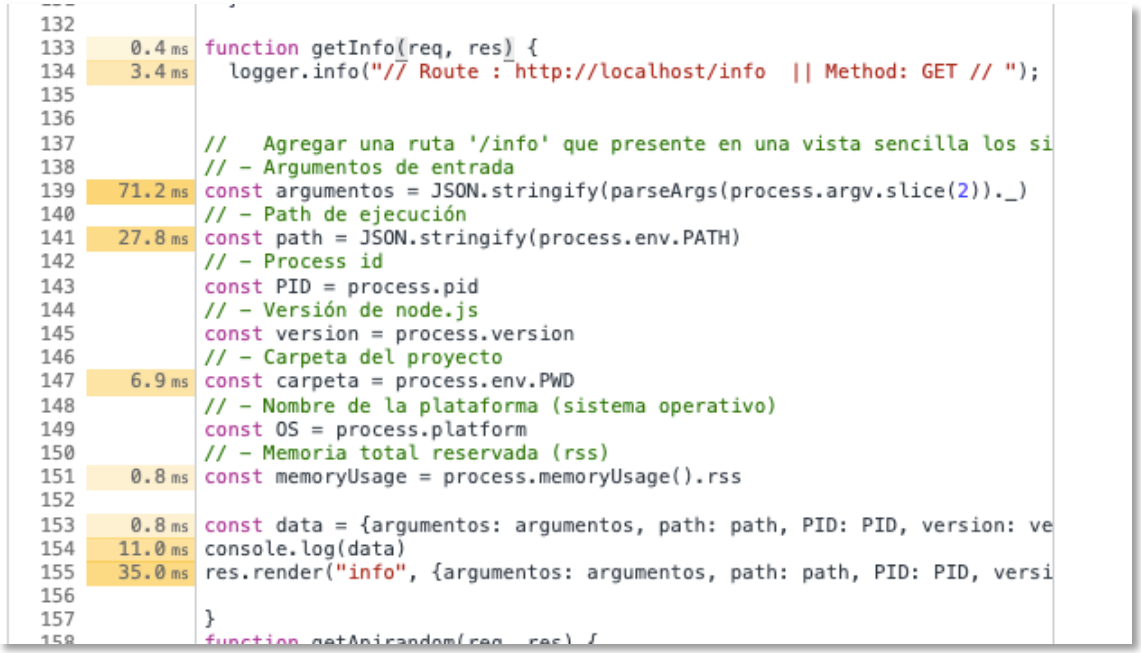
```
Desafio-14 > Performance > result_fork_sinCL.txt

Metrics for period to: 19:30:10(-0300) (width: 1.718s)

http.codes.200: ..... 193
http.request_rate: ..... 95/sec
http.requests: ..... 147
http.response_time:
  min: ..... 174
  max: ..... 585
  median: ..... 415.8
  p95: ..... 572.6
  p99: ..... 584.2
http.responses: ..... 193
vusers.completed: ..... 46
vusers.failed: ..... 0
vusers.session_length:
  min: ..... 8679.9
  max: ..... 9184.7
  median: ..... 9047.6
  p95: ..... 9230.4
  p99: ..... 9230.4

All VUs finished. Total time: 11 seconds
```

```
Desafio-14 > Performance > result_fork_conCL.txt

Metrics for period to: 15:27:50(-0300) (width: 4.221s)

http.codes.200: ..... 690
http.codes.302: ..... 656
http.request_rate: ..... 314/sec
http.requests: ..... 1296
http.response_time:
  min: ..... 11
  max: ..... 470
  median: ..... 175.9
  p95: ..... 391.6
  p99: ..... 432.7
http.responses: ..... 1346
vusers.completed: ..... 50
vusers.failed: ..... 0
vusers.session_length:
  min: ..... 13598.2
  max: ..... 14429
  median: ..... 14048.5
  p95: ..... 14332.3
  p99: ..... 14332.3

All VUs finished. Total time: 16 seconds
```

Resultados autocannon SIN console.log //// Resultados autocannon CON console.log

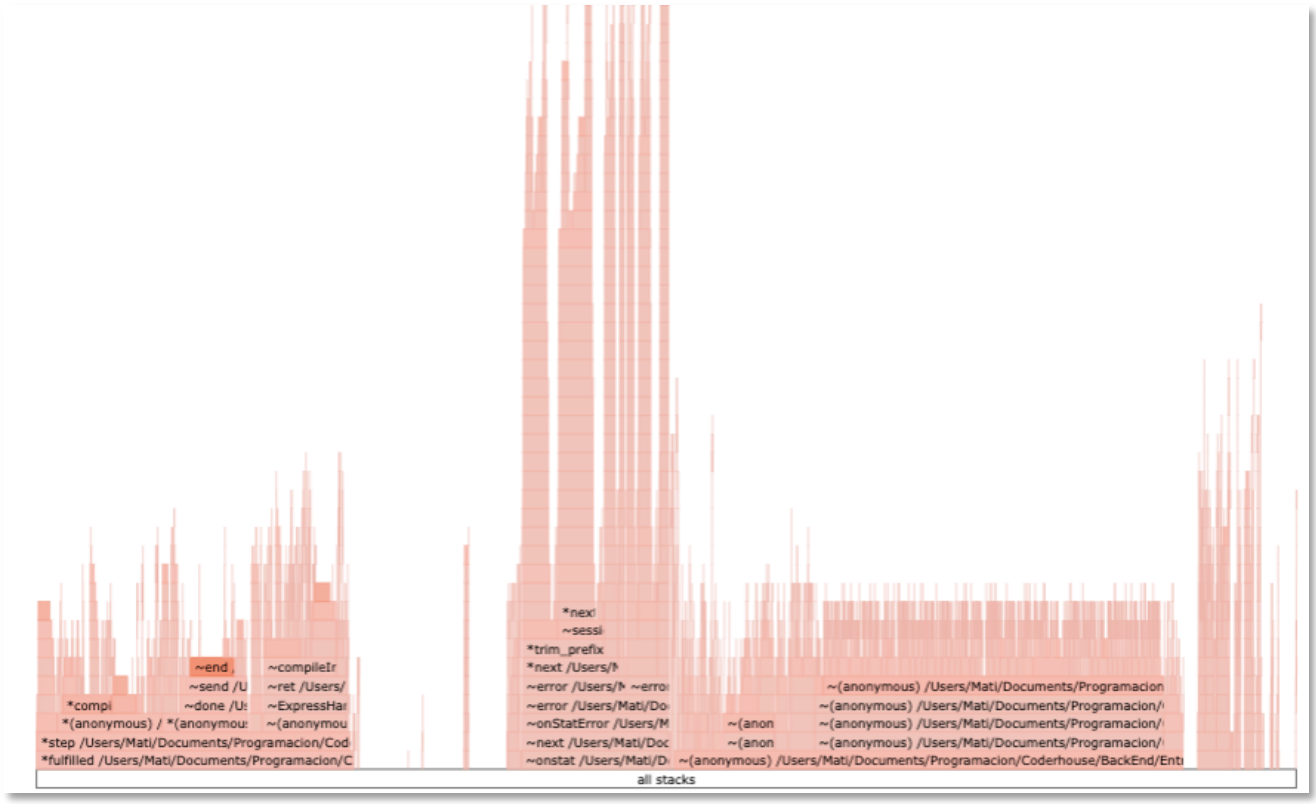
```
133          function getInfo(req, res) {
134    3.2 ms    logger.info("// Route : http://localhost/info  || Method: GET // ");
135
136
137          //   Agregar una ruta '/info' que presente en una vista sencilla los si
138          // - Argumentos de entrada
139   30.6 ms const argumentos = JSON.stringify(parseArgs(process.argv.slice(2))._)
140          // - Path de ejecución
141   23.7 ms const path = JSON.stringify(process.env.PATH)
142          // - Process id
143          const PID = process.pid
144          // - Versión de node.js
145    0.2 ms const version = process.version
146          // - Carpeta del proyecto
147    3.5 ms const carpeta = process.env.PWD
148          // - Nombre de la plataforma (sistema operativo)
149          const OS = process.platform
150          // - Memoria total reservada (rss)
151    2.0 ms const memoryUsage = process.memoryUsage().rss
152
153          // const data = {argumentos: argumentos, path: path, PID: PID, version:
154          // console.log(data)
155   20.8 ms res.render("info", {argumentos: argumentos, path: path, PID: PID, versi
156
157          }
158          function getApirandom(req, res) {
```

Tiempos menos performantes según devtools Chrome SIN console.log

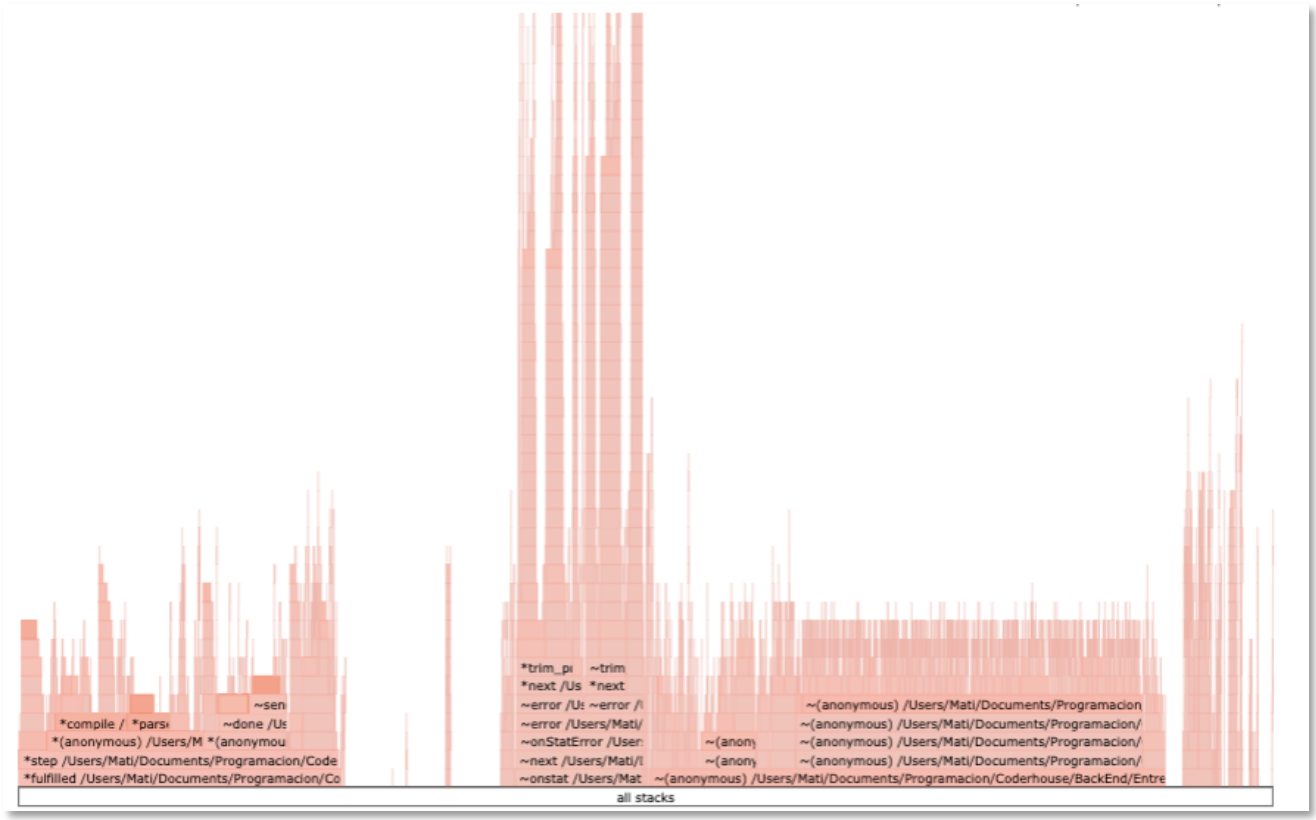


```
132
133    0.4 ms function getInfo(req, res) {
134    3.4 ms   logger.info("// Route : http://localhost/info  || Method: GET // ");
135
136
137          //   Agregar una ruta '/info' que presente en una vista sencilla los si
138          // - Argumentos de entrada
139   71.2 ms const argumentos = JSON.stringify(parseArgs(process.argv.slice(2))._)
140          // - Path de ejecución
141   27.8 ms const path = JSON.stringify(process.env.PATH)
142          // - Process id
143          const PID = process.pid
144          // - Versión de node.js
145          const version = process.version
146          // - Carpeta del proyecto
147    6.9 ms const carpeta = process.env.PWD
148          // - Nombre de la plataforma (sistema operativo)
149          const OS = process.platform
150          // - Memoria total reservada (rss)
151    0.8 ms const memoryUsage = process.memoryUsage().rss
152
153    0.8 ms const data = {argumentos: argumentos, path: path, PID: PID, version: ve
154   11.0 ms console.log(data)
155   35.0 ms res.render("info", {argumentos: argumentos, path: path, PID: PID, versi
156
157          }
158          function getApirandom(req, res) {
```

Tiempos menos performantes según devtools Chrome CON console.log



Analisis 0x SIN console.log



Analisis 0x CON console.log

Conclusión:

1)Con Autocannon se observa un 50% más de tiempo en terminar las tareas cuando incluyen los console.log.

2)Con la herramienta devtools del inspector de Chrome se observan mayores tiempos de carga mucho mayores en la función de getInfo, que es la que trabaja el endpoint analizado. Incluso se observa que esa función es la que más tiempo lleva en total cuando funciona con console.log y cuando no, la función que más tiempo lleva es el logger.

3)No se observa demasiada diferencia con el análisis de 0x. Una mínima diferencia en cuanto al "calor" de algunos procesos (mayor calor en la función con console.log).