





## Consigna 1

### gzip



Sin gzip, el documento en la ruta /info pesa 1kB:



Name	Status	Type	Initiator	Size	Time	Waterfall
 info	200	document	Other	1.0 kB	13 ms	

Con gzip, también pesa 1kB:

Name	Status	Type	Initiator	Size	Time	Waterfall
 info	200	document	Other	1.0 kB	8 ms	

Esto se debe a que gzip no comprime archivos por debajo de 1kB. Añadiendo texto aleatorio para aumentar el tamaño del documento a 7kB, se puede observar que se da una compresión con gzip hasta los 753B:

Name	Status	Type	Initiator	Size	Time	Waterfall
 info	200	document	Other	7.0 kB	22 ms	

Name	Status	Type	Initiator	Size	Time	Waterfall
 info	200	document	Other	753 B	49 ms	

## Consigna 2

- - prof

Sin loguear en consola los datos de /info:

[Summary]:			
ticks	total	nonlib	name
110	5.7%	6.1%	JavaScript
1673	86.5%	92.8%	C++
177	9.2%	9.8%	GC
131	6.8%		Shared libraries
20	1.0%		Unaccounted

Logeando por consola los datos de /info:

[Summary]:			
ticks	total	nonlib	name
85	4.5%	4.9%	JavaScript
1636	87.1%	94.1%	C++
126	6.7%	7.2%	GC
139	7.4%		Shared libraries
18	1.0%		Unaccounted

Como se puede observar las diferencias son mínimas, esto se debe a que el texto de la ruta /info ocupa apenas 1kB (como se puede ver en la consigna 1) por lo que no impacta significativamente la performance del servidor el hecho de logearlo o no por consola.

## - - inspect

Se observa que el impacto de logear por consola la información de la ruta /info resulta en una demora de 1.7ms:

```
97      let folder = process.cwd();
98      0.3 ms    let cores = os.cpus().length;
99      1.7 ms    console.log(args+so+nodeVer+rss+execPath+pId+folder+cores)
00      1.2 ms    res.render("info", {args, so, nodeVer, rss, execPath, pId, folder, cores})
01    });
```

## 0x & Autocannon

El mayor proceso bloqueante se corresponde con el renderizado de Handlebars, con un 21% de tiempo en el stack.

