

Despliegue de una aplicación MEAN en el IaaS de la ULL

Samuel Martín Morales

October 10, 2023

Contents

1	Introducción	2
2	Despliegue de una aplicación MEAN de prueba	3
3	Clonado de la máquina virtual Backend	7
4	Configuración del Proxy como sistema de balanceo de carga	10
5	Conclusión	11

Chapter 1

Introducción

Para esta tercera práctica de la asignatura *Sistemas y Tecnologías Web*, se solicita el despliegue de una aplicación MEAN que permite comprobar la estructura de servidores que ha sido implementada en prácticas anteriores de la asignatura.

Tras esto, se solicita la creación de un nuevo servidor backend que se trate de una copia o de un clon del servidor backend que existía de manera previa en la infraestructura implantada.

Finalmente, tras la realización de estos pasos anteriores, se pide configurar la infraestructura para que el servidor *Nginx* pueda realizar un balanceo de carga de las peticiones realizadas hacia dicho servidor Proxy, el cual se encargará de realizar la distribución de las peticiones entre los dos servidores backend que existen en la infraestructura, permitiendo por tanto, un balanceo de carga.

Chapter 2

Despliegue de una aplicación MEAN de prueba

Para el despliegue de una aplicación MEAN para la comprobación de la infraestructura implantada en prácticas anteriores, se hace uso de la siguiente aplicación de prueba. Dicha aplicación, necesita del cambio de ciertos aspectos para que se pueda ejecutar de manera correcta en nuestra infraestructura.

Pero, antes de comenzar con los cambios necesarios dentro del código de la aplicación clonada, se debe de realizar la configuración de la máquina backend, para que, permita que *Visual Studio Code* pueda abrir la carpeta que contiene los distintos ficheros de la aplicación, dentro de la máquina virtual que se encuentra alojada en el IaaS de la ULL.

En un comienzo, se obtiene la clave rsa que se encuentra dentro de nuestra máquina personal, para poder realizar la conexión SSH con la máquina virtual que se encuentra alojada en el IaaS de la ULL. Para ello, se ejecuta el siguiente comando:

```
ssh-keygen -t rsa -b 4096 -C "  
samuel.martin.morales@estudiante.  
ull.es"
```

A continuación, se accede a la máquina virtual la cual se quiere configurar para que permita la conexión SSH desde la máquina personal, y se ejecutan los siguientes comandos:

```
mkdir ~/.ssh  
touch authorized_keys  
sudo nano authorized_keys
```

Dentro de dicho fichero denominado como *authorized_keys*, añadimos la clave rsa generada de manera previa dentro de la máquina personal, y, tras todo esto ya se puede realizar la conexión de la máquina virtual con visual studio code para poder editar código dentro de este. Todo esto comentado anteriormente, queda reflejado en la siguiente imagen 2.1.

```
..mentos/mean-stack-example Backend-Server:~/ssh ..@MacBook-Air-Smartin:~
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
manpath: can't set the locale; make sure $LC_* and $LANG are correct

usuario@Backend-Server ~ (0.032s)
ls
mean-stack-example
  10   app.use(express.static('public'));
  11
  12   app.get('/', function (req, res) {
usuario@Backend-Server:~ (0.29s)
  13     res.send('Hello World!');
  14   res.render('index', { title: 'Express' });
  15   });
ls -la
total 48
drwx----- 7 usuario usuario 4096 Oct  6 10:21 .
drwxr-xr-x 5 root root 4096 Sep 28 11:24 ..
-rw----- 1 usuario usuario 1008 Oct  6 10:17 .bash_history
-rw-r--r-- 1 usuario usuario 220 Sep 28 11:10 .bash_logout
-rw-r--r-- 1 usuario usuario 3723 Oct  6 10:00 .bashrc
-rw-r--r-- 1 usuario usuario 0 Oct  6 09:56 .cloud-locale-test.skip
drwxr-xr-x 3 usuario usuario 4096 Oct  6 10:03 .npm
drwxr-xr-x 5 usuario usuario 4096 Oct  6 10:01 .nvm
-rw-r--r-- 1 usuario usuario 807 Sep 28 11:10 .profile
drwxr-xr-x 2 usuario usuario 4096 Oct  6 10:17 .ssh: 'Chuchu' {}
-rw-r--r-- 1 usuario usuario 0 Oct  6 09:52 .sudo_as_admin_successful
drwxr-xr-x 5 usuario usuario 4096 Oct  6 10:21 .vscode-server
-rw-r--r-- 1 usuario usuario 294 Oct  6 10:21 .wget-hsts
drwxr-xr-x 6 usuario usuario 4096 Oct  6 10:03 mean-stack-example
  16     res.send('Got a GET request');
  17   });
  18 }
usuario@Backend-Server ~ (0.034s)
  19   <br/>'
cd .ssh/
  20   });
  21 }
usuario@Backend-Server ~/.ssh (0.04s)
  22   /dog', function (req, res) {
ls
  23     res.sendFile(path.join(__dirname, 'public/dog.jpg'));
authorized_keys
  24   });
  25 }
usuario@Backend-Server ~/.ssh
  26   server = app.listen(8080, ip.address(), function () {
Type '#' for AI command suggestions
  27   = server.address().address
```

Figure 2.1: Configuración de la máquina virtual para que se pueda editar código haciendo uso de Visual Studio Code.

Una vez configurado Visual Studio Code para poder editar el código necesario, se realiza la configuración del fichero `.env` dentro del `server` de la aplicación MEAN. Dentro de dicho fichero, se modifica el `ATLAS-URI` existente, de tal manera que, permita conectarnos desde el server del backend hasta la base de datos implantada mediante *MongoDB* dentro del servidor de base de datos que se ha instalado de manera previa en prácticas anteriores. Con todo esto anteior, se puede observar la siguiente imagen 2.2.

```
ATLAS_URI=mongodb://Samuel-Test:tibYDKQ8@172.16.13.78:27017/test
```

Figure 2.2: Configuración del fichero `.env`.

Como se puede observar en la variable de entorno `ATLAS-URI`, la estructura de la ruta es la siguiente:

```
mongodb://mongodb://<username>:<password>@<ip_direction>:<port>/<database_name>
```

Otra de las configuraciones que se deben de hacer para la correcta ejecución de la aplicación MEAN suministrada, se produce dentro del fichero denominado *package.json* dentro del cliente de la aplicación. En este fichero, se debe de configurar uno de los scripts de inicio del cliente, de tal manera que, se encuentre escuchando la interfaz de red de nuestro servidor backend, de forma que, cuando el servidor Proxy realice un *Proxy_Pass* hacia el servidor backend, este pueda escuchar las peticiones que se le realizan desde el servidor Proxy. Con todo esto comentado, se puede observar la siguiente imagen 2.3 dónde se muestra la correcta configuración del script de inicio del cliente de la aplicación MEAN, de manera que funcione para la infraestructura implementada.

```
{
  "name": "client",
  "version": "0.0.0",
  "author": "Stanimira Vlaeva",
  "license": "Apache 2.0",
  > Depurar
  "scripts": {
    "ng": "ng",
    "start": "ng serve --host 172.16.12.78 --port 3000",
    "build": "ng build",
    "watch": "ng build --watch --configuration development",
    "lint": "ng lint",
    "test": "ng lint"
  },
}
```

Figure 2.3: Configuración del script de inicio del cliente.

Tras estas configuraciones realizadas, se puede iniciar la instalación de las dependencias necesarias:

```
npm install
```

Una vez instaladas las dependencias necesarias, se puede iniciar la aplicación MEAN de prueba, para ello, se ejecuta el siguiente comando:

```
npm start
```

Finalmente, se puede observar el correcto funcionamiento de la aplicación MEAN de prueba, en las siguientes imágenes 2.4, 2.5.

```
[0] Server running at http://localhost:5200...
[1] - Generating browser application bundles (phase: setup)...
[1]   TypeScript compiler options "target" and "useDefineForClassFields" are set to "ES2022" and "false" respectively by the Angular CLI.
[1]   es use the Browserslist configuration. For more information, see https://angular.io/guide/build#configuring-browser-compatibility
[1]   ✓ Browser application bundle generation complete.
[1]
[1] Initial Chunk Files | Names | Raw Size
[1] vendor.js | vendor | 2.50 MB
[1] polyfills.js | polyfills | 317.03 kB
[1] styles.css, styles.js | styles | 209.39 kB
[1] main.js | main | 37.93 kB
[1] runtime.js | runtime | 6.51 kB
[1]
[1] | Initial Total | 3.06 MB
[1]
[1] Build at: 2023-10-09T15:22:11.503Z - Hash: 4dd968e3107dcb7b - Time: 19821ms
[1]
[1] ** Angular Live Development Server is listening on 172.16.12.78:3000, open your browser on http://172.16.12.78:3000/ **
[1]
[1]
[1] ✓ Compiled successfully.
```

Figure 2.4: Funcionamiento de la aplicación MEAN de prueba dentro de la terminal de la máquina Backend.

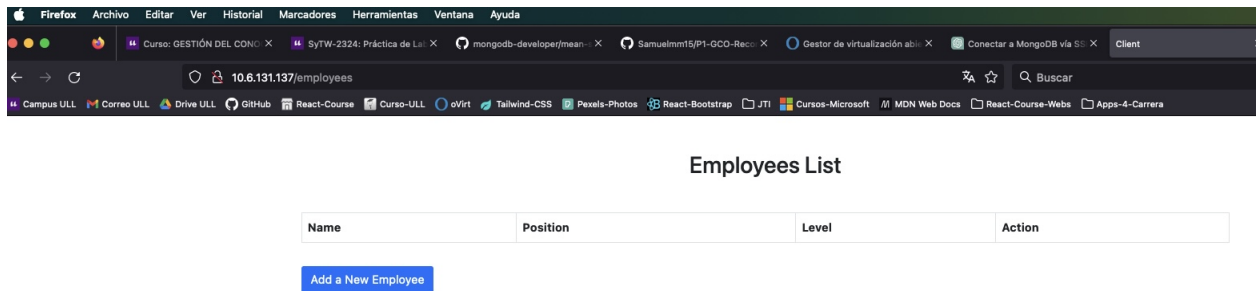


Figure 2.5: Funcionamiento de la aplicación MEAN, desde cualquier navegador.

Chapter 3

Clonado de la máquina virtual Backend

Una vez se ha comprobado el correcto funcionamiento de la infraestructura implantada en prácticas anteriores gracias al despliegue de la aplicación MEAN de prueba, se procede a la clonación del servidor backend existente de manera previa en la infraestructura. Para ello, se realiza una copia de la máquina virtual que se encuentra alojada en el IaaS de la ULL, para ello, se observa en la siguiente imagen 3.1 dónde se puede ver la nueva estructura de la infraestructura, junto con el establecimiento de la distintas direcciones ip que van a ser instauradas dentro de las distintas interfaces de red de dicha máquina.

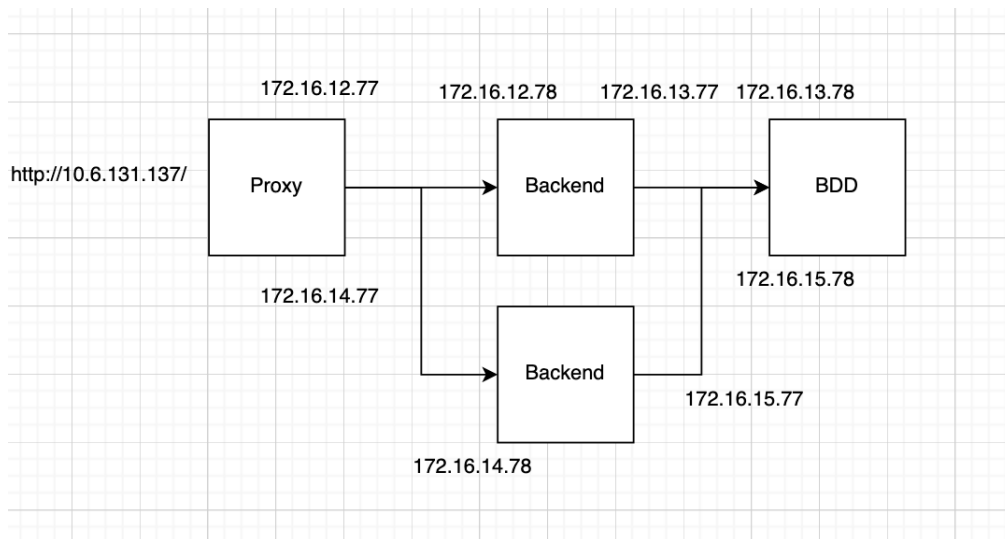


Figure 3.1: Estructura de la nueva configuración de infraestructura.

Teniendo en cuenta todo esto anterior, se realiza la creación de la nueva máquina virtual, la cuál, poseerá dos nuevas interfaces de red, una de ellas correspondiente con la *DOCINT4* (`172.16.14.77/172.16.14.78`) y la *DOCINT3* (`172.16.15.77/172.16.15.78`). Antendiendo a esto, se puede observar el siguiente fichero de configuración de interfaces 3.2 y la comprobación del funcionamiento de las interfaces de red de la nueva máquina clonada 3.3 y 3.4.


```
This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# Máquinas virtuales
auto ens4
iface ens4 inet dhcp

auto ens7
iface ens7 inet static
address 172.16.14.78
netmask 255.255.255.0

auto ens8
iface ens8 inet static
address 172.16.15.77
netmask 255.255.255.0

# The primary network interface
allow-hotplug ens3
# iface ens3 inet dhcp
```

Figure 3.2: Configuración de interfaces de red de la nueva máquina clonada.

```
usuario@Proxy-Server ~ (0.043s)
tracert 172.16.14.78
tracert to 172.16.14.78 (172.16.14.78), 30 hops max, 60 byte packets
 1 172.16.14.78 (172.16.14.78) 0.240 ms 0.204 ms 0.185 ms

usuario@Proxy-Server ~ (2.296s)
ping 172.16.14.78
PING 172.16.14.78 (172.16.14.78) 56(84) bytes of data.
 64 bytes from 172.16.14.78: icmp_seq=1 ttl=64 time=0.222 ms
 64 bytes from 172.16.14.78: icmp_seq=1 ttl=64 time=0.223 ms (DUP!)
 64 bytes from 172.16.14.78: icmp_seq=1 ttl=64 time=0.223 ms (DUP!)
 64 bytes from 172.16.14.78: icmp_seq=2 ttl=64 time=0.283 ms
 64 bytes from 172.16.14.78: icmp_seq=2 ttl=64 time=0.283 ms (DUP!)
 64 bytes from 172.16.14.78: icmp_seq=2 ttl=64 time=0.316 ms (DUP!)
 64 bytes from 172.16.14.78: icmp_seq=2 ttl=64 time=0.316 ms (DUP!)
 64 bytes from 172.16.14.78: icmp_seq=3 ttl=64 time=0.310 ms
 64 bytes from 172.16.14.78: icmp_seq=3 ttl=64 time=0.310 ms (DUP!)
 64 bytes from 172.16.14.78: icmp_seq=3 ttl=64 time=0.310 ms (DUP!)
 64 bytes from 172.16.14.78: icmp_seq=3 ttl=64 time=0.310 ms (DUP!)
^C
--- 172.16.14.78 ping statistics ---
 3 packets transmitted, 3 received, +9 duplicates, 0% packet loss, time 2051ms
 rtt min/avg/max/mdev = 0.222/0.277/0.316/0.040 ms
```

Figure 3.3: Comprobación del funcionamiento de las interfaces de red de la nueva máquina clonada.

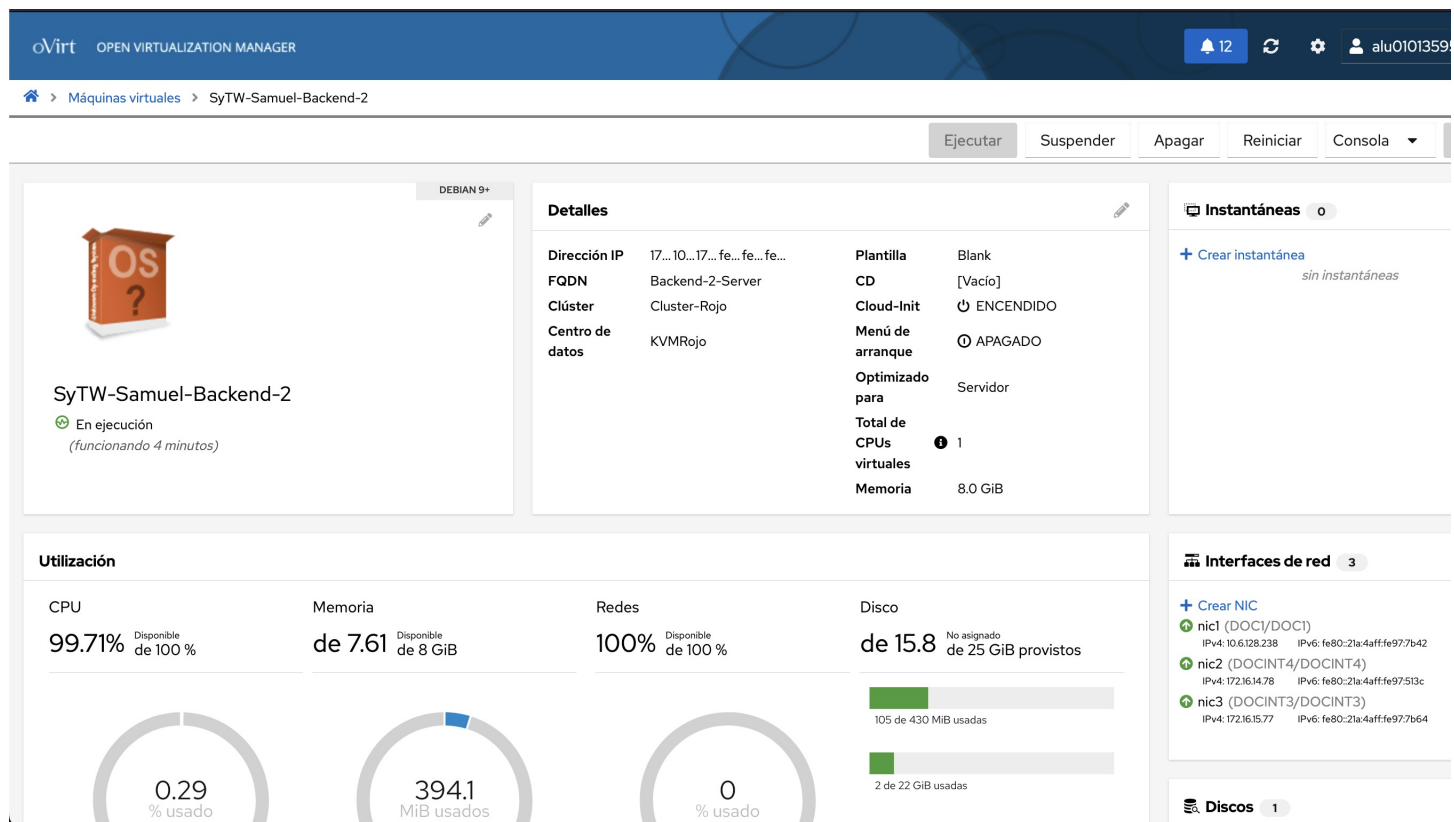


Figure 3.4: Comprobación del funcionamiento de las interfaces de red de la nueva máquina clonada.

Finalmente, tras la configuración de manera correcta de las distintas interfaces de red de la nueva máquina clonada, se procede a la instalación de los distintos paquetes que fueron instalados de manera previa, para ello, se puede visualizar el informe de la práctica 2 de la asignatura *Sistemas y Tecnologías Web*.

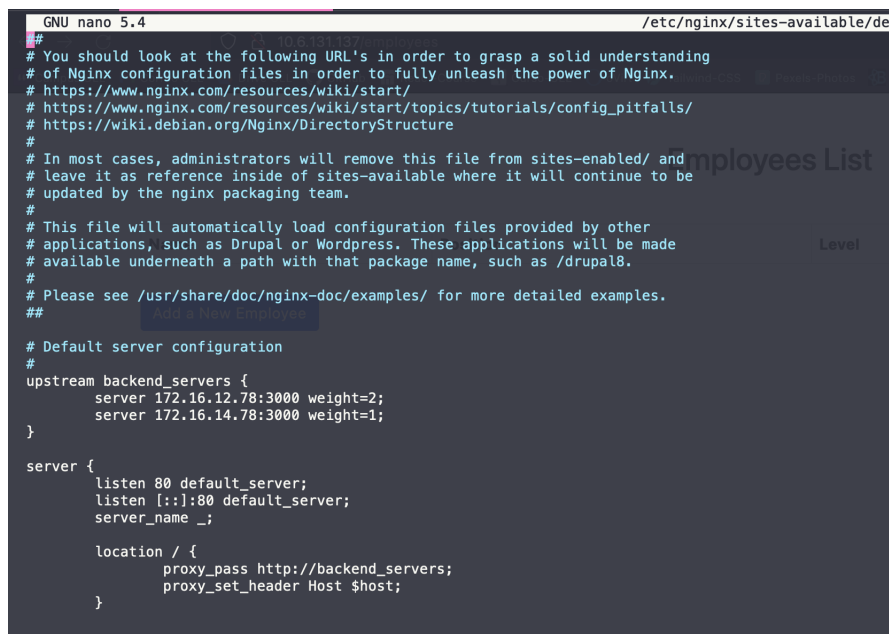
Chapter 4

Configuración del Proxy como sistema de balanceo de carga

Una vez se ha realizado la clonación de la máquina virtual que se encuentra alojada en el IaaS de la ULL, se procede a la configuración del servidor Proxy para que este pueda realizar un balanceo de carga entre los dos servidores backend que existen en la nueva infraestructura. Para ello, se realiza la configuración de *Nginx*, de manera que pueda redistribuir peticiones entre los dos servidores backend que existen en esta.

En este caso, se configura el fichero *default*, de manera que pueda realizar el balanceo de la carga de forma que el doble de las peticiones irá para el servidor backend con interfaz de red *172.16.12.78*, mientras que el otro servidor backend con interfaz de red *172.16.14.78* tendrá un peso de 1.

Todo esto comentado anteriormente, se puede observar en la siguiente configuración del fichero de configuración *default* de *Nginx* 4.1.

A screenshot of a terminal window showing the configuration of the Nginx default file. The window title is 'GNU nano 5.4' and the file path is '/etc/nginx/sites-available/default'. The configuration includes comments about Nginx configuration files and URLs, instructions for administrators, and a default server configuration. The server configuration includes an upstream block for backend servers and a server block for listening on port 80. The upstream block defines two servers: 172.16.12.78:3000 with weight 2, and 172.16.14.78:3000 with weight 1. The server block listens on port 80 and proxies requests to the backend servers.

```
GNU nano 5.4 /etc/nginx/sites-available/default
#
# You should look at the following URL's in order to grasp a solid understanding
# of Nginx configuration files in order to fully unleash the power of Nginx.
# https://www.nginx.com/resources/wiki/start/
# https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from sites-enabled/ and
# leave it as reference inside of sites-available where it will continue to be
# updated by the nginx packaging team.
#
# This file will automatically load configuration files provided by other
# applications, such as Drupal or Wordpress. These applications will be made
# available underneath a path with that package name, such as /drupal8.
#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
##

# Default server configuration
#
upstream backend_servers {
    server 172.16.12.78:3000 weight=2;
    server 172.16.14.78:3000 weight=1;
}

server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name _;

    location / {
        proxy_pass http://backend_servers;
        proxy_set_header Host $host;
    }
}
```

Figure 4.1: Configuración del fichero default de Nginx para la implementación del sistema de balanceo de carga.

Chapter 5

Conclusión

Para finalizar, se puede concluir que la realización de esta práctica ha sido de gran utilidad para poder comprender el funcionamiento de un sistema de balanceo de carga, así como la configuración de un servidor Proxy para que este pueda realizar un balanceo de carga entre los distintos servidores backend que existen en la infraestructura. Además, se ha podido comprobar el correcto funcionamiento de la infraestructura implantada en prácticas anteriores, gracias al despliegue de una aplicación MEAN de prueba. Es por ello que, esta tercera práctica de la asignatura me ha servido como aprendizaje para poder buscarme la vida para comprender el funcionamiento del código y de la aplicación implantada por otro programador como es la aplicación MEAN usada para la comprobación del funcionamiento de la infraestructura.

Para finalizar, el establecimiento de un sistema de balanceo de carga también me permite comprender un poco más sobre cómo funciona de manera real las infraestructuras de servidores que se encuentran en la actualidad, ya que, en la mayoría de los casos, se suele implementar un sistema de balanceo de carga entre los distintos servidores backend que existen, de manera que se pueda realizar una distribución de las peticiones que se realizan hacia el servidor Proxy, entre los distintos servidores de la infraestructura, permitiendo por tanto, un balanceo de carga.