**DESAFIO ENTREGABLE 12 – Balance de carga**

**Primera parte:**

Se incorporó en la constante args un parámetro con el alias “m” para referenciar al mode de ejecución del servidor. De esta manera se recibirá como param si el servidor debe iniciarse en modo fork o en modo cluster.

Por consola ingresar los siguientes comandos:

**-pm2 start src/server.js -- -p 8081 -m cluster**

**-pm2 start src/server2.js -- -p 8082 -m fork**

De esta forma server.js se estará ejecutado en el puerto 8081 en modo cluster y el server2.js en el puerto 8082 en modo fork (1 solo proceso).

Para probarlo corrí en un cmd el siguiente comando

**-tasklist**

y de esta manera pude ver todos los procesos de node corriendo en paralelo en mi sistema operativo.

Luego en el navegador abrí las siguientes rutas:

**-http://localhost:3000/api/login** 🡪 para poder ver el servidor.js ejecutado en modo cluster

Y por otro lado:

**-http://localhost:3000/api2/login** 🡪 para poder ver el servidor2.js ejecutado en modo fork.

Esto es gracias a la configuración de **nginx** que me permite declarar que todas las peticiones que lleguen a **“/api”** vayan al servidor cluster corriendo en el puerto **8081** y las que llegan a **“api2”** vayan al servidor corriendo en el puerto **8082**.

La configuración de **nginx.conf** usada es la siguiente:

events {

}

http {

    include mime.types;

    default\_type application/octet-stream;

    upstream node\_app\_cluster {

        server 127.0.0.1:8081;

    }

    upstream node\_app {

        server 127.0.0.1:8082;

    }

    server {

        listen 80;

        server\_name nginx\_node;

        root ../NginxNode/public;

        location /api/ {

            proxy\_pass http://node\_app\_cluster;

        }

        location /api2 {

            proxy\_pass http://node\_app;

        }

    }

}

**Segunda parte:**

Para la segunda parte del ejercicio creé nuevos archivos en la raíz de mi proyecto que se llamaron **server2a.js**, **server2b.js**, **server2c.js** y **server2d.js**. Todos estos tienen el ruteo que comienza con **/api2**.

Luego cambio la configuración de **nginx.conf** para que se vea así:

events {

}

http {

    include mime.types;

    default\_type application/octet-stream;

    upstream node\_app\_cluster {

        server 127.0.0.1:8082;

        server 127.0.0.1:8083;

        server 127.0.0.1:8084;

        server 127.0.0.1:8085;

    }

    upstream node\_app {

        server 127.0.0.1:8081;

    }

    server {

        listen 80;

        server\_name nginx\_node;

        root ../NginxNode/public;

        location /api/ {

            proxy\_pass http://node\_app;

        }

        location /api2 {

            proxy\_pass http://node\_app\_cluster;

        }

    }

}

Esta configuración es para que todas las consultas a mi servidor que empiecen con la ruta **/api** se dirijan al puerto **8081** de mi pc, mientras que todas las consultas que le llegue con la ruta **/api2** se redistribuyan entre los distintos puertos **8082-8083-8084-8085**.

El comando de consola usado para ejecutar esto es:

-**pm2 start src/server.js -- -p 8081 -m fork && pm2 start src/server2a.js -- -p 8082 -m fork && pm2 start src/server2b.js -- -p 8083 -m fork && pm2 start src/server2c.js -- -p 8084 -m fork && pm2 start src/server2d.js -- -p 8085 -m fork**

Para probarlo abrimos en el navegador la siguiente ruta:

[**http://localhost/api/info**](http://localhost/api/info) --> allí veremos que el puerto usado es el 8081

Luego probamos esta otra ruta:

[**http://localhost/api2/info**](http://localhost/api2/info) --> al actualizar esta ventana veremos que el puerto que nos muestra esta vista de info irá cambiando entre los puertos configurados en la configuración de nginx.