**Primera parte:**

Se agrega dentro de la constante args el parámetro con el alias “m” (haciendo referencia al mode de ejecución del servidor) para poder recibir el params que indicara al programa si debe iniciarse en modo fork o en modo cluster.

Por consola (bash) 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 el server.js 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 corro en un cmd el siguiente comando y alli veo todos los procesos de node corriendo en paralelo en mi sistema operativo:

**tasklist**

Luego en el navegador abro las siguientes rutas:

**localhost/api/login** para poder ver el servidor.js ejecutado en modo cluster

Y potr otro lado:

**localhost/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](http://node_app_cluster/);

        }

        location /api2 {

            proxy\_pass [http://node\_app](http://node_app/);

        }

    }

}

-------------------------------------------------------------------------------------------------------------------

**Segunda parte:**

Para la segunda parte del ejercicio creo nuevos archivos en la raíz de mi proyecto que se llamaran **server2.js**, **server3.js**, **server4.js** y **server5.js**. Todos estos tienen una ruta 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](http://node_app/);

        }

        location /api2 {

            proxy\_pass [http://node\_app\_cluster](http://node_app_cluster/);

        }

    }

}

Esta configuración es para que todas las consultas al servidor que empiecen con la ruta **/api** se dirijan al puerto **8081**, 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/server2.js -- -p 8082 -m fork && pm2 start src/server3.js -- -p 8083 -m fork && pm2 start src/server4.js -- -p 8084 -m fork && pm2 start src/server5.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 ira cambiando entre los puertos configurados en la configuración de nginx.