

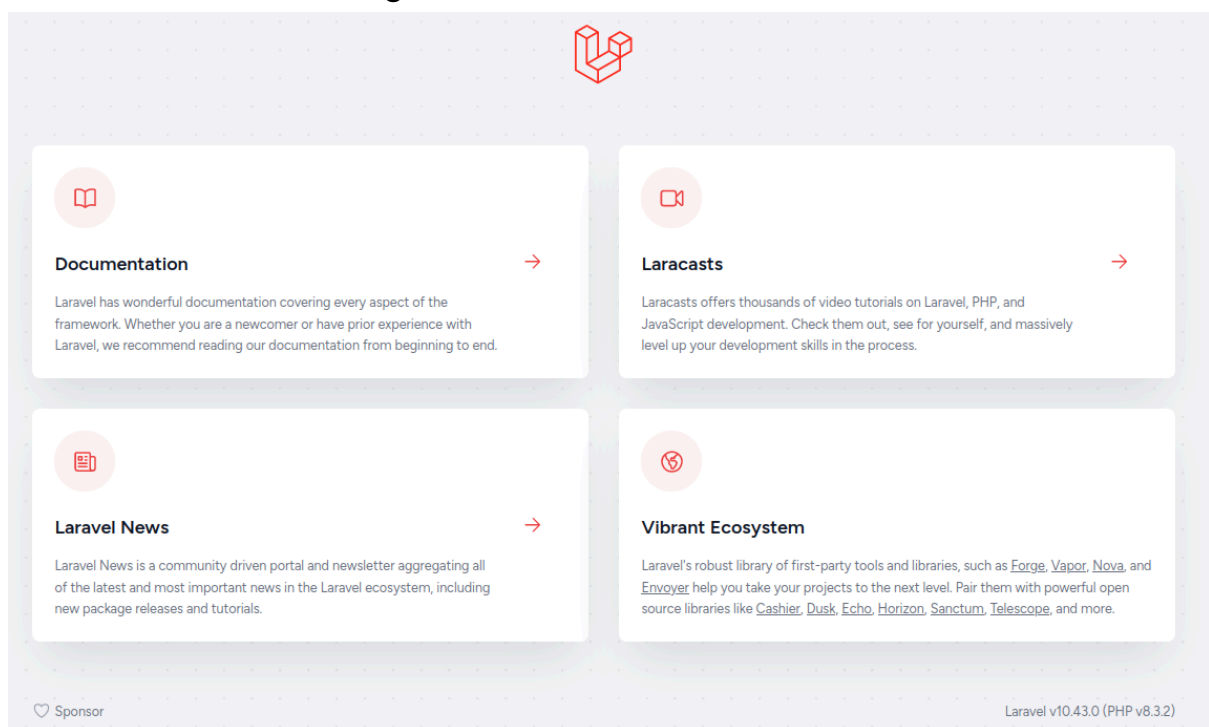
Actividad Bloque I

1. Manual de configuración

Necesitaremos la aplicación contenida en practica.7z. La abriremos en una terminal, y para levantarla, introduciremos:

- docker-compose build
- docker-compose up

Nos debería mostrar algo así:



(ventana inicial de una aplicación de Laravel)

- Para añadir funcionalidad de balanceo de carga / proxy reverso, deberemos añadir la directiva proxy_pass (y la dirección nueva) a nuestra configuración de nginx (laravel-app/nginx/conf.d/app.conf):

```
location / {  
    try_files $uri $uri/ /index.php?$query_string;  
    gzip_static on;  
    proxy_pass http://google.com;  
}
```

- Para manejar varias páginas en una instancia de nginx, deberemos añadir otra directiva location con nuestra ruta relativa a la aplicación. Un ejemplo sencillo se vería así:

```
location /hola {  
    root /var/www;  
    index index.php index.html index.htm;  
}
```

 index.html X

laravel-app > hola >  index.html >  h1

1

2 `<h1>Hola Mundo!</h1>`

localhost/hola/ X +

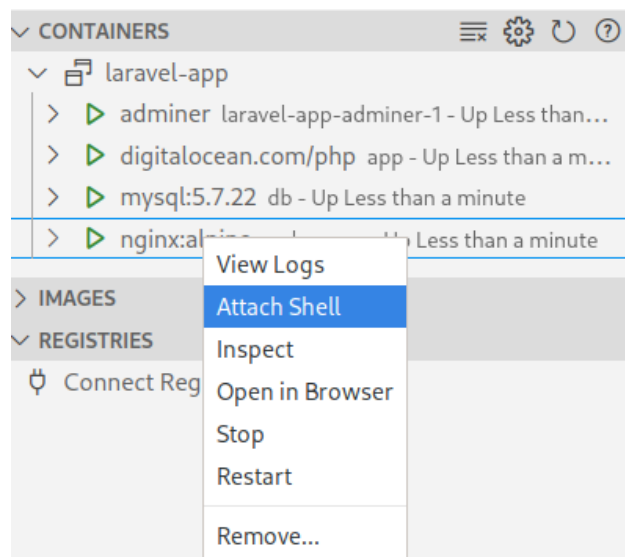
← → ↻   localhost/hola/

Hola Mundo!

- Para proteger partes de la aplicación con contraseña, deberemos añadir a nuestra configuración de nginx las siguientes directivas:

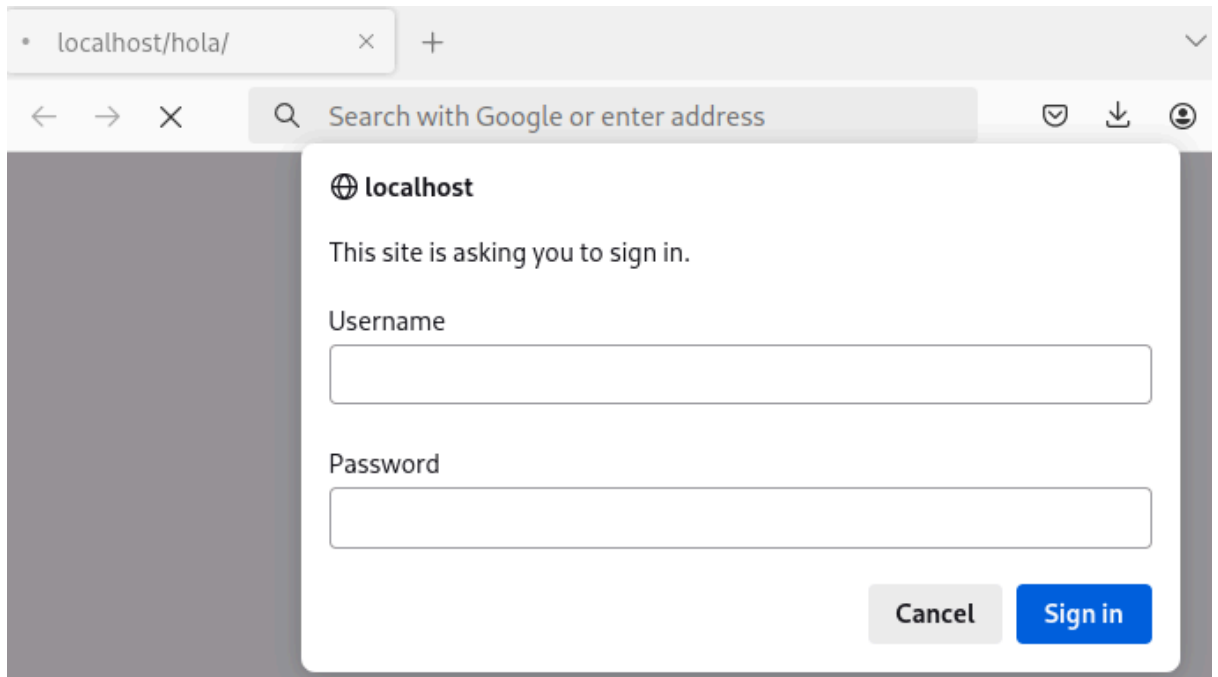
```
location /hola {
    root /var/www;
    index index.php index.html index.htm;
    auth_basic "Área restringida";
    auth_basic_user_file /etc/pwd/.htpasswd;
}
```

Al arrancar el contenedor, deberemos abrir una shell e instalar el programa apache2-utils (para generar el archivo de contraseña), crear el archivo de contraseña en la ruta indicada y reiniciar el servicio:



```
/ # apk add apache2-utils
fetch https://dl-cdn.alpinelinux.org/alpine/v3.18/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.18/community/x86_64/APKINDEX.tar.gz
(1/3) Installing apr (1.7.4-r0)
(2/3) Installing apr-util (1.6.3-r1)
(3/3) Installing apache2-utils (2.4.58-r0)
Executing busybox-1.36.1-r5.trigger
OK: 46 MiB in 67 packages
/ # mkdir /etc/pwd/
/ # htpasswd -c /etc/pwd/.htpasswd rodrigo
New password:
Re-type new password:
Adding password for user rodrigo
/ # nginx -s reload
2024/02/11 16:57:47 [notice] 53#53: signal process started
/ #
```

Cuando accedamos a nuestra ruta con autorización, se nos pedirá contraseña:



The image shows a web browser window with a single tab titled "localhost/hola/". The address bar contains the text "Search with Google or enter address". A sign-in dialog box is displayed in the foreground, titled "localhost" with a globe icon. The dialog contains the text "This site is asking you to sign in." followed by two input fields: "Username" and "Password". At the bottom right of the dialog are two buttons: "Cancel" and "Sign in".

localhost/hola/

Search with Google or enter address

localhost

This site is asking you to sign in.

Username

Password

Cancel Sign in

- Para manejar peticiones HTTPS, lo primero es añadir el puerto seguro en la sección del servidor web en el archivo docker-compose.yml:

```
ports:
  - '80:80'
  - '443:443' # Puerto SSL
```

También añadiremos el enlace a la carpeta de nuestra aplicación donde guardaremos los certificados (en el docker-compose.yml, en la sección del servidor):

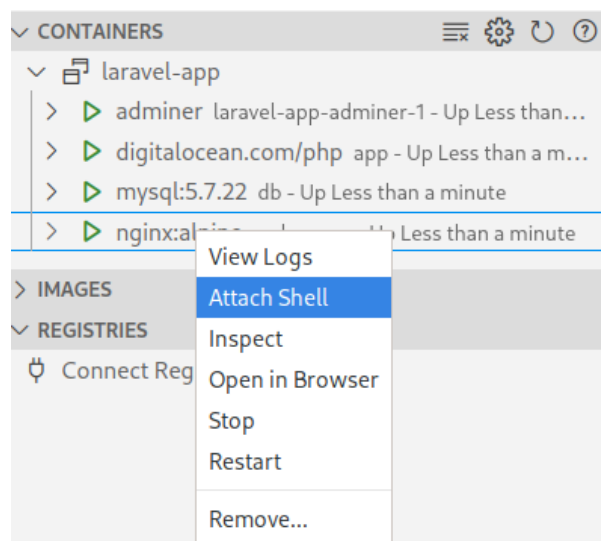
```
volumes:
  - './:/var/www'
  - './nginx/conf.d:/etc/nginx/conf.d/'
  - './nginx/ssl:/etc/nginx/ssl/ # Carpeta de certificados
```

Tendremos que añadir el puerto en la propia configuración del nginx, mediante estas directivas en la sección server:

```
listen 443 ssl; # Puerto SSL
index index.php index.html;
error_log /var/log/nginx/error.log;
access_log /var/log/nginx/access.log;
root /var/www/public;
```

```
ssl_certificate /etc/nginx/ssl/laravel-app.com.crt; # El propio certificado
ssl_certificate_key /etc/nginx/ssl/laravel-app.com.key; # La llave del certificado
```

Ahora instalaremos las aplicaciones para generar los susodichos certificados, mediante una consola del contenedor:



```
/ # apk add apache2-utils openssl
(1/4) Installing apr (1.7.4-r0)
(2/4) Installing apr-util (1.6.3-r1)
(3/4) Installing apache2-utils (2.4.58-r0)
(4/4) Installing openssl (3.1.4-r5)
Executing busybox-1.36.1-r5.trigger
OK: 46 MiB in 68 packages
```

Para generar el certificado y su clave, deberemos, en la shell del contenedor, ejecutar estos comandos:

```
/ # openssl genrsa 2048 > laravel-app.com.key
```

Para crear la susodicha clave

```
/ # openssl req -new -key laravel-app.com.key -out laravel-app.com.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Madrid
Locality Name (eg, city) []:Madrid
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Prácticas EIP
Organizational Unit Name (eg, section) []:Desarrollo Fullstack
Common Name (e.g. server FQDN or YOUR name) []:Rodrigo
Email Address []:rodrigo@emailfalso.com

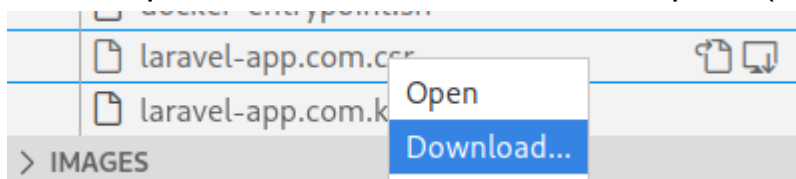
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:12345678
An optional company name []:Prácticas EIP
```

Para crear la petición de firma

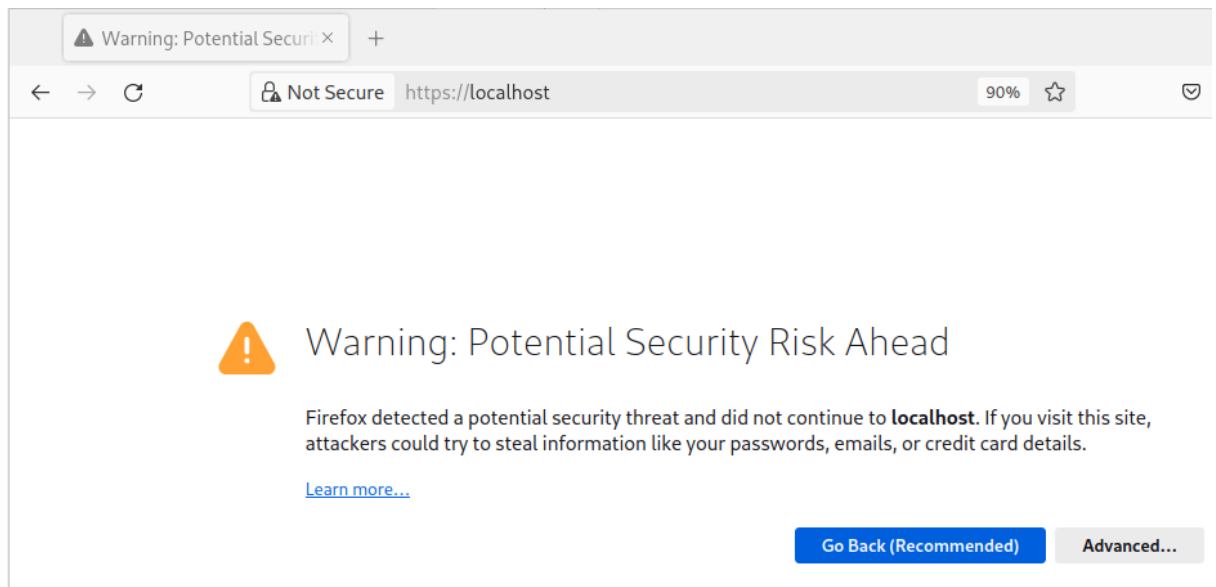
```
/ # openssl x509 -signkey laravel-app.com.key -in laravel-app.com.csr -req -days 365
-out laravel-app.com.crt
Certificate request self-signature ok
subject=C = ES, ST = Madrid, L = Madrid, O = Pr\C3\83\C2\A1cticas EIP, OU = Desarrol
lo Fullstack, CN = Rodrigo, emailAddress = rodrigo@emailfalso.com
```

Para crear el mismo certificado

Ahora descargaremos los archivos *.crt y *.key y los colocaremos en la carpeta marcada en el docker-compose (nginx/ssl):



Al reiniciar el servicio y entrar a la URL podemos ver que nos dice que tenemos un certificado inválido, ya que no está firmado por una organización oficial. Podemos ver nuestros datos:



Certificate

Rodrigo	
Subject Name	
Country	ES
State/Province	Madrid
Locality	Madrid
Organization	PrÁcticas EIP
Organizational Unit	Desarrollo Fullstack
Common Name	Rodrigo
Email Address	rodrigo@emailfalso.com

2. Balanceo de carga

Primero, añadiremos las direcciones del backend a nuestro archivo app.conf:


```
upstream back {  
    server 192.168.1.33:1111;  
    server 192.168.1.33:1112;  
    server 192.168.1.33:1113;  
}
```


Para cambiar de modo del balanceo de carga, hay que añadir directivas dependiendo del susodicho. Si no se pone nada, va por defecto a round robin (resuelve direcciones de una en una en orden cíclico).

Añadiremos la directiva `least_conn`, para que el servidor resuelva las direcciones en el servidor con menos carga:

```
upstream back {  
    least_conn;  
    server 192.168.1.33:1111;  
    server 192.168.1.33:1112;  
    server 192.168.1.33:1113;  
}
```

Para probar el funcionamiento, crearemos una aplicación de Express sencilla:



```
'use strict';  
const express = require('express');  
// Constants  
const PORT = 777;  
const HOST = '0.0.0.0';  
// App  
const app = express();  
app.get('/back', (req, res) => {  
    res.status(200).json('Esto es una aplicacion final desde el servidor 3');  
});  
app.listen(PORT, HOST, () => {  
    console.log(`Running on http://${HOST}:${PORT}`);  
});
```

Ahora construiremos tres imágenes diferentes de la aplicación Express mediante:

- `docker build . -t nodeserver-1`

Y los asignaremos a los puertos mediante:

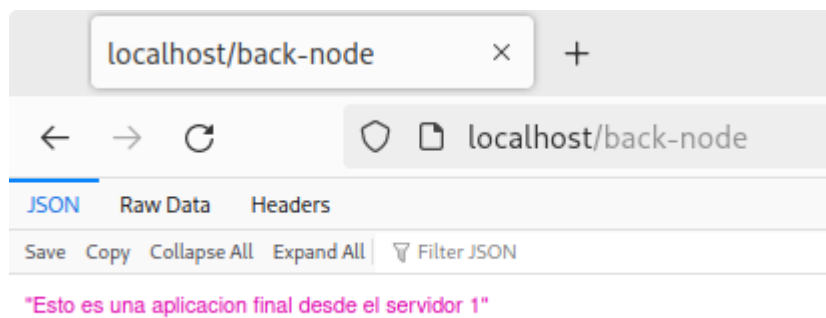
- `docker run -d --rm -it --name server1 -p 1111:777 nodeserver-1`

Para probarlo en la aplicación, añadiremos un location a nuestra configuración de nginx para ir a nuestro back:

```
location /back-node {  
    proxy_pass http://back/back;  
}
```

Referenciando a los servidores de upstream balanceados.

Ahora, al llegar a localhost/back-node recibimos:



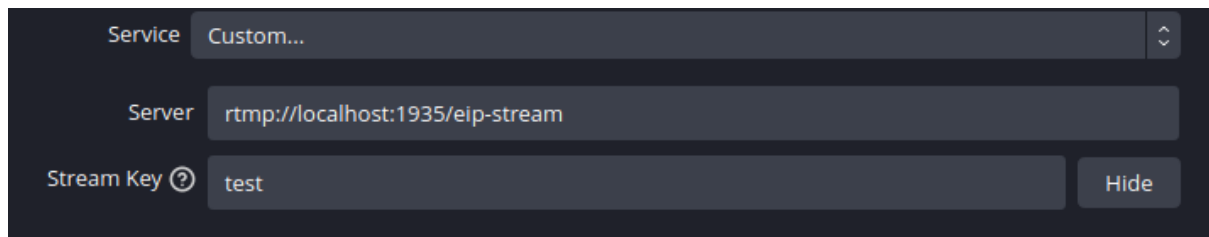
3. Funcionalidad adicional

Una funcionalidad adicional que no hemos visto es el streaming nativo. Para ello, deberemos cambiar nuestra imagen de nginx a la incluida en el archivo comprimido, ya que se necesita un módulo propietario no incluido en la imagen nginx:alpine.

Hay que añadir una sección nueva a nuestra configuración de nginx, fuera de la sección server:

```
rtmp {  
    server {  
        listen 1935;  
        application eip-stream {  
            live on;  
            record off;  
        }  
    }  
}
```

También necesitaremos un programa para hacer el propio directo. En mi caso, estoy usando OBS. En la configuración de directo, deberemos añadir la configuración a nuestro servidor de nginx:



Service Custom...

Server rtmp://localhost:1935/eip-stream

Stream Key ? test Hide

La clave de stream es para indicar qué stream se está viendo, ya que un servidor puede alojar varios directos.

Al hacer el directo, y conectarnos con un visualizador de vídeo como VLC, podremos ver el directo:

