Nombre: NICOLAS CANO COLORADO 18/11/2025

TRABAJO PRACTICO DAW

INDICE

PLANTEAMIENTO DE ARQUITECTURA Y SELECCIÓN DE TECNOLOGIAS

Se desarrolla una arquitectura simple cliente-servidor. Para ello, se utilizará lo siguiente:

Tecnologías:

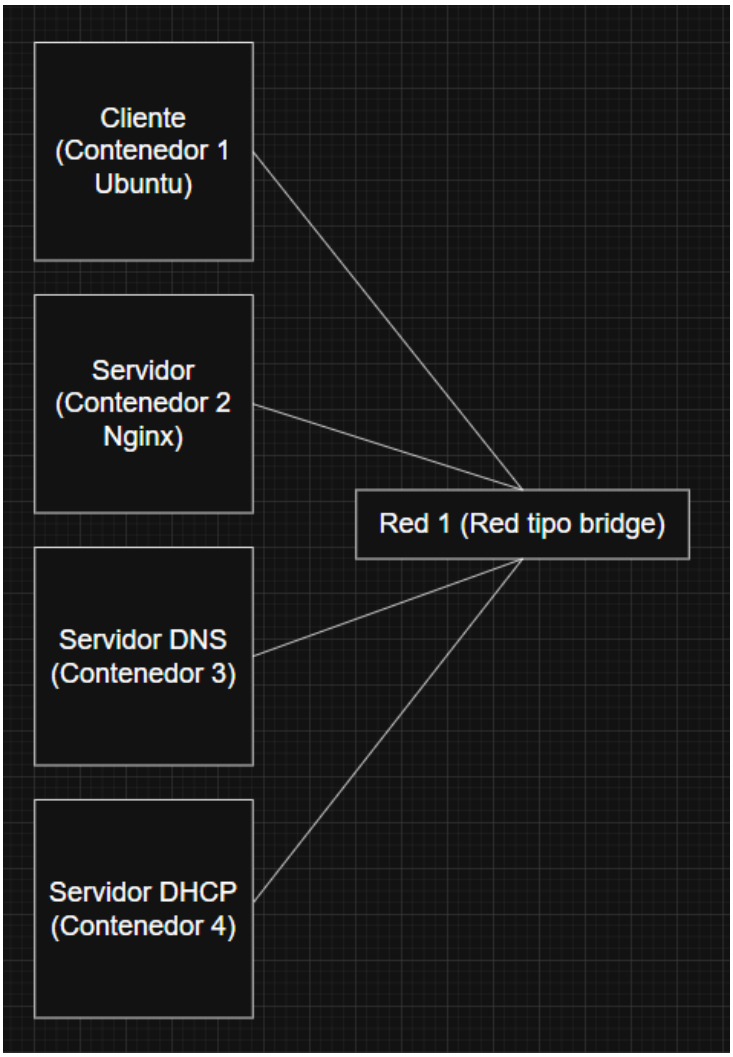
- Docker: Gestión de contenedores y redes, además de su red Docker bridge.
- Linux (Ubuntu): Sistema operativo para el lado de cliente
- Nginx: Servidor web del lado de servidor
- Protocolos de red: TCP/IP, DNS, DHCP y HTTP/HTTPS para usar sus respectivos servicios

Servicios:

- DNS: Para resolver los nombres de redes
- DHCP: Asignara automáticamente la mayoría de IPs, máscaras y DNS. (Simulado, ya que el bridge de Docker ya lo hace por defecto)
- Servidor web: Para ser el host de la página web.
- Cliente en red: Para visitar la página web.

Diagrama:

El diagrama de la arquitectura planteada sería el siguiente:



Se representa en el diagrama una arquitectura Cliente-Servidor donde hay presente adicionalmente un servidor DNS y un servidor DHCP, sumando un total de 4 contenedores. Todas estas se conectan por una red llamada Red 1, la cual es de tipo bridge.

Tanto el servidor DNS como el DHCP están presentes como simulados, pero como Docker ya tiene su propio DNS y el bridge de Docker ya hace de DHCP, no hace falta configurar otros.

Creación de los contenedores

Creación de cliente:

```
docker run -it --name cliente ubuntu bash
```

Se ha instalado en el las actualizaciones, net-tools, ping y nano. (apt update, apt upgrade, apt install net-tools, apt install nano, apt install iputils-ping y crul)

Creación de servidor:

```
docker run -d --name servidor_web nginx
```

Al igual que con el anterior, también se le descargan y las actualizaciones y herramientas. Se le aplicara “detached” de antemano. (Las mismas que cliente)

Creación de servidor DNS:

```
docker run -d --name servidor_dns internetsystemsconsortium/bind9:9.18
```

Este se también permanecerá “detached”. Además, debido a que Docker ya viene con su propio DNS, este no se configurara. Este contenedor es meramente representativo. No requiere instalaciones como en los otros.

Creación de servidor DHCP:

```
docker run -d --name servidor_dhcp debian:bookworm-slim sleep infinity
```

Al igual que con el dns, permanecerá “detached” y no se configurará. Estará meramente representativo, pues la red bridge de Docker ya se encarga de ello. También se le instalan las actualizaciones y herramientas correspondientes (Las mismas que cliente)

Creación de la red:

```
docker network create red
```

Conexión de contenedores a red

```
docker network connect red cliente
docker network connect red servidor_web
docker network connect red servidor_dns
docker network connect red servidor_dhcp
```

“Ping” entre contenedores

- Desde cliente

```
docker exec -it cliente bash
```

```
root@b0d333949fc7:/# ping servidor_web
PING servidor_web (172.24.0.3) 56(84) bytes of data.
64 bytes from servidor_web.red (172.24.0.3): icmp_seq=1 ttl=64 time=0.132 ms
64 bytes from servidor_web.red (172.24.0.3): icmp_seq=2 ttl=64 time=0.142 ms
64 bytes from servidor_web.red (172.24.0.3): icmp_seq=3 ttl=64 time=0.118 ms
64 bytes from servidor_web.red (172.24.0.3): icmp_seq=4 ttl=64 time=0.168 ms
64 bytes from servidor_web.red (172.24.0.3): icmp_seq=5 ttl=64 time=0.137 ms
^C
--- servidor_web ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4058ms
rtt min/avg/max/mdev = 0.118/0.139/0.168/0.016 ms
```

```
root@b0d333949fc7:/# ping servidor_dns
PING servidor_dns (172.24.0.4) 56(84) bytes of data.
64 bytes from servidor_dns.red (172.24.0.4): icmp_seq=1 ttl=64 time=0.127 ms
64 bytes from servidor_dns.red (172.24.0.4): icmp_seq=2 ttl=64 time=0.060 ms
64 bytes from servidor_dns.red (172.24.0.4): icmp_seq=3 ttl=64 time=0.119 ms
64 bytes from servidor_dns.red (172.24.0.4): icmp_seq=4 ttl=64 time=0.061 ms
64 bytes from servidor_dns.red (172.24.0.4): icmp_seq=5 ttl=64 time=0.137 ms
^C
--- servidor_dns ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4092ms
rtt min/avg/max/mdev = 0.060/0.100/0.137/0.033 ms
root@b0d333949fc7:/#
```

```
root@b0d333949fc7:/# ping servidor_dhcp
PING servidor_dhcp (172.24.0.5) 56(84) bytes of data.
64 bytes from servidor_dhcp.red (172.24.0.5): icmp_seq=1 ttl=64 time=0.110 ms
64 bytes from servidor_dhcp.red (172.24.0.5): icmp_seq=2 ttl=64 time=0.139 ms
64 bytes from servidor_dhcp.red (172.24.0.5): icmp_seq=3 ttl=64 time=0.587 ms
64 bytes from servidor_dhcp.red (172.24.0.5): icmp_seq=4 ttl=64 time=0.158 ms
64 bytes from servidor_dhcp.red (172.24.0.5): icmp_seq=5 ttl=64 time=0.157 ms
^C
--- servidor_dhcp ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4076ms
rtt min/avg/max/mdev = 0.110/0.230/0.587/0.179 ms
root@b0d333949fc7:/#
```

- Desde servidor_web

```
docker exec -it servidor_web bash
```

```
root@2f343054374b:/# ping cliente
PING cliente (172.24.0.2) 56(84) bytes of data.
64 bytes from cliente.red (172.24.0.2): icmp_seq=1 ttl=64 time=0.080 ms
64 bytes from cliente.red (172.24.0.2): icmp_seq=2 ttl=64 time=0.081 ms
64 bytes from cliente.red (172.24.0.2): icmp_seq=3 ttl=64 time=0.091 ms
64 bytes from cliente.red (172.24.0.2): icmp_seq=4 ttl=64 time=0.129 ms
64 bytes from cliente.red (172.24.0.2): icmp_seq=5 ttl=64 time=0.130 ms
^C
--- cliente ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4064ms
rtt min/avg/max/mdev = 0.080/0.102/0.130/0.022 ms
```

```
root@2f343054374b:/# ping servidor_dns
PING servidor_dns (172.24.0.4) 56(84) bytes of data.
64 bytes from servidor_dns.red (172.24.0.4): icmp_seq=1 ttl=64 time=0.152 ms
64 bytes from servidor_dns.red (172.24.0.4): icmp_seq=2 ttl=64 time=0.063 ms
64 bytes from servidor_dns.red (172.24.0.4): icmp_seq=3 ttl=64 time=0.130 ms
64 bytes from servidor_dns.red (172.24.0.4): icmp_seq=4 ttl=64 time=0.126 ms
64 bytes from servidor_dns.red (172.24.0.4): icmp_seq=5 ttl=64 time=0.101 ms
^C
--- servidor_dns ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4103ms
rtt min/avg/max/mdev = 0.063/0.114/0.152/0.030 ms
```

```
root@2f343054374b:/# ping servidor_dhcp
PING servidor_dhcp (172.24.0.5) 56(84) bytes of data.
64 bytes from servidor_dhcp.red (172.24.0.5): icmp_seq=1 ttl=64 time=0.095 ms
64 bytes from servidor_dhcp.red (172.24.0.5): icmp_seq=2 ttl=64 time=0.145 ms
64 bytes from servidor_dhcp.red (172.24.0.5): icmp_seq=3 ttl=64 time=0.106 ms
64 bytes from servidor_dhcp.red (172.24.0.5): icmp_seq=4 ttl=64 time=0.139 ms
64 bytes from servidor_dhcp.red (172.24.0.5): icmp_seq=5 ttl=64 time=0.130 ms
64 bytes from servidor_dhcp.red (172.24.0.5): icmp_seq=6 ttl=64 time=0.169 ms
^C
--- servidor_dhcp ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5110ms
rtt min/avg/max/mdev = 0.095/0.130/0.169/0.024 ms
```

- Desde servidor_dns

```
docker exec -it servidor_dns sh
```

```
/ # ping cliente
PING cliente (172.24.0.2): 56 data bytes
64 bytes from 172.24.0.2: seq=0 ttl=64 time=0.084 ms
64 bytes from 172.24.0.2: seq=1 ttl=64 time=0.146 ms
64 bytes from 172.24.0.2: seq=2 ttl=64 time=0.098 ms
64 bytes from 172.24.0.2: seq=3 ttl=64 time=0.174 ms
64 bytes from 172.24.0.2: seq=4 ttl=64 time=0.063 ms
^C
--- cliente ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.063/0.113/0.174 ms
/ #
```

```
/ # ping servidor_web
PING servidor_web (172.24.0.3): 56 data bytes
64 bytes from 172.24.0.3: seq=0 ttl=64 time=0.096 ms
64 bytes from 172.24.0.3: seq=1 ttl=64 time=0.171 ms
64 bytes from 172.24.0.3: seq=2 ttl=64 time=0.186 ms
64 bytes from 172.24.0.3: seq=3 ttl=64 time=0.154 ms
64 bytes from 172.24.0.3: seq=4 ttl=64 time=0.163 ms
^C
--- servidor_web ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.096/0.154/0.186 ms
```

```
/ # ping servidor_dhcp
PING servidor_dhcp (172.24.0.5): 56 data bytes
64 bytes from 172.24.0.5: seq=0 ttl=64 time=0.128 ms
64 bytes from 172.24.0.5: seq=1 ttl=64 time=0.167 ms
64 bytes from 172.24.0.5: seq=2 ttl=64 time=0.067 ms
64 bytes from 172.24.0.5: seq=3 ttl=64 time=0.161 ms
64 bytes from 172.24.0.5: seq=4 ttl=64 time=0.351 ms
^C
--- servidor_dhcp ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.067/0.174/0.351 ms
```

- Desde servidor_dhcp

```
docker exec -it servidor_dhcp bash
```

```
64 bytes from cliente.red (172.24.0.2): icmp_seq=1 ttl=64 time=0.106 ms
64 bytes from cliente.red (172.24.0.2): icmp_seq=2 ttl=64 time=0.137 ms
64 bytes from cliente.red (172.24.0.2): icmp_seq=3 ttl=64 time=0.128 ms
64 bytes from cliente.red (172.24.0.2): icmp_seq=4 ttl=64 time=0.134 ms
64 bytes from cliente.red (172.24.0.2): icmp_seq=5 ttl=64 time=0.111 ms
^C
--- cliente ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
```

```
root@d2299a6e4902:/# ping servidor_web
PING servidor_web (172.24.0.3) 56(84) bytes of data.
64 bytes from servidor_web.red (172.24.0.3): icmp_seq=1 ttl=64 time=0.120 ms
64 bytes from servidor_web.red (172.24.0.3): icmp_seq=2 ttl=64 time=0.052 ms
64 bytes from servidor_web.red (172.24.0.3): icmp_seq=3 ttl=64 time=0.126 ms
64 bytes from servidor_web.red (172.24.0.3): icmp_seq=4 ttl=64 time=0.133 ms
64 bytes from servidor_web.red (172.24.0.3): icmp_seq=5 ttl=64 time=0.140 ms
^C
--- servidor_web ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4051ms
rtt min/avg/max/mdev = 0.052/0.114/0.140/0.031 ms
```

```
root@d2299a6e4902:/# ping servidor_dns
PING servidor_dns (172.24.0.4) 56(84) bytes of data.
64 bytes from servidor_dns.red (172.24.0.4): icmp_seq=1 ttl=64 time=0.111 ms
64 bytes from servidor_dns.red (172.24.0.4): icmp_seq=2 ttl=64 time=0.182 ms
64 bytes from servidor_dns.red (172.24.0.4): icmp_seq=3 ttl=64 time=0.134 ms
64 bytes from servidor_dns.red (172.24.0.4): icmp_seq=4 ttl=64 time=0.137 ms
64 bytes from servidor_dns.red (172.24.0.4): icmp_seq=5 ttl=64 time=0.127 ms
^C
--- servidor_dns ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4021ms
rtt min/avg/max/mdev = 0.111/0.138/0.182/0.023 ms
```

Comprobación que cliente pueda pedir recursos a servidor

Instalamos curl

```
root@b0d333949fc7:/# apt install curl
```

Usamos curl y nos da la pagina por defecto del servidor_web

```
root@b0d333949fc7:/# curl http://servidor_web
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Instalación y configuración de servidores web con Docker

Creación de contenedores

- Se crea la red de antemano

```
docker network create red_servidor
```

- Se crea un cliente. Se le instalan las actualizaciones y herramientas correspondientes (apt update, apt upgrade, apt install net-tools, apt install nano, apt install iputils-ping)

```
docker run -it --name cliente2 ubuntu bash
```

- Se crea el servidor. También se le instala todo lo necesario. (apt update, apt upgrade, apt install net-tools, apt install nano, apt install iputils-ping)

```
docker run -it --name servidorNginx nginx bash
```

Cambio de puertos del servidor de 80 a 90

Se enciende el contenedor

```
docker start servidorNginx
```

Se entra en el contenedor

```
docker exec -it servidorNginx bash
```

Se busca default.conf, donde está la configuración del puerto.

```
root@2c0b6abc697b:/etc/nginx/conf.d# nano default.conf
```

Se cambia el puerto de 80 a 90 (Con nano)

```
server {
    listen       80;
    server_name  localhost;
```

```
server {
    listen       90;
    server_name  localhost;
```

Se guarda, se sale del contenedor con exit y se crea la imagen (servidornginx90)

```
docker commit servidorNginx servidornginx90
```

Se mata y borra el contenedor y se vuelve a crear, pero con la imagen configurada y los puertos 90

```
docker kill servidorNginx
docker rm servidorNginx
docker run -it --name servidorNginx -p 90:90 servidornginx90 bash
```

Se puede ver como Docker ya lo tiene configurado por el puerto 90

```
docker ps
        PORTS                                       NAMES
econds   0.0.0.0:90->90/tcp, [::]:90->90/tcp   servidorNginx
```

Se entra una vez mas al contenedor con “Docker exec” y se reinicia para aplicar cambios.

```
nginx -s reload
```

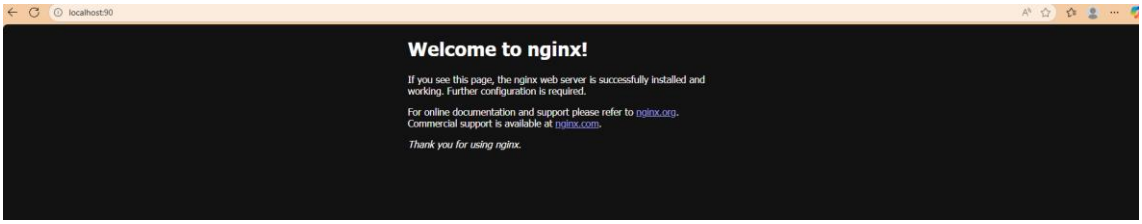
Disponibilidad y acceso

Revisamos que funcione de 2 formas:

- Con curl:

```
root@2c0b6abc697b:/# curl http://localhost:90
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

- Y desde el navegador:



Índex personalizado

Vamos al índex y lo editamos con nano:

```
root@2c0b6abc697b:/usr/share/nginx/html# nano index.html
```

Se ve así sin editar

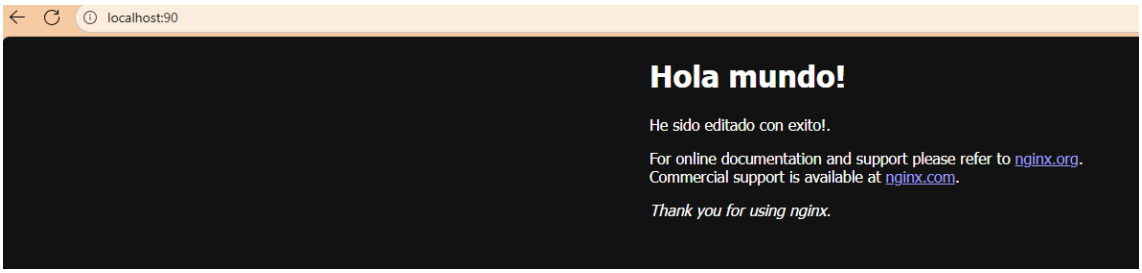
```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Y editado quedo:

```
  GNU nano 8.4
<!DOCTYPE html>
<html>
<head>
<title>Soy una aplicacion, y estoy despegando!!!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Hola mundo!</h1>
<p>He sido editado con exito!.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Después de guardar, hacemos otro reinicio y miramos el navegador a ver si funciono

```
nginx -s reload
```



Efectivamente, se ven los cambios.

Configuraciones de seguridad

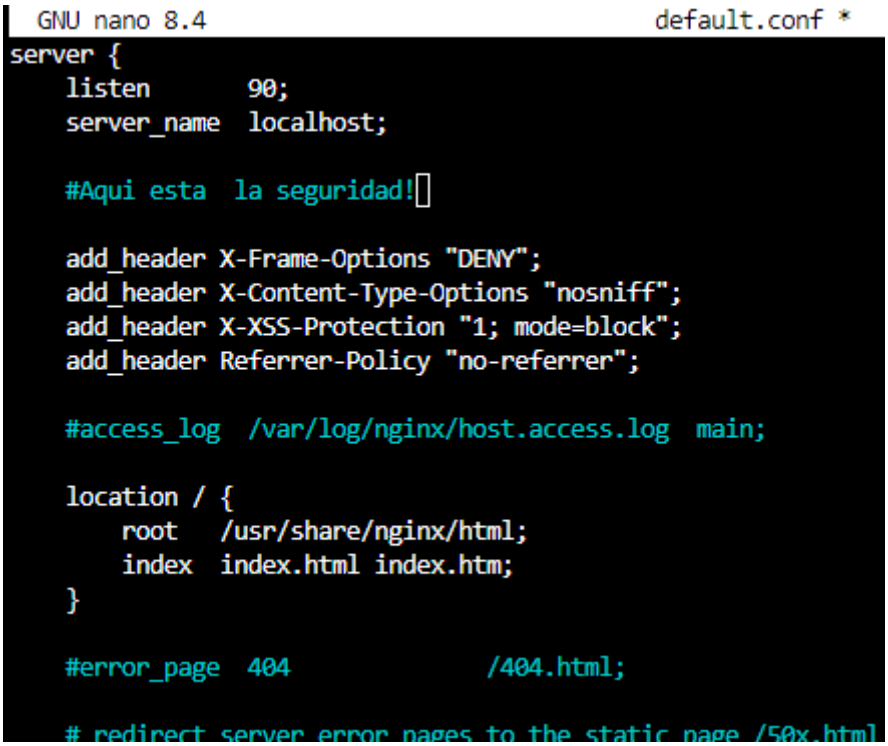
- Primero, volvemos a donde se configuraron los puertos

```
root@2c0b6abc697b:/etc/nginx/conf.d# nano default.conf
```

- Debajo de server, ponemos las configuraciones. Se veía asi

```
server {
    listen       90;
    server_name  localhost;

    #access_log  /var/log/nginx/host.access.log  main;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
    }

    #error_page  404              /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

    # proxy the PHP scripts to Apache listening on 127.0.0.1:80
```

Y ahora se ve así

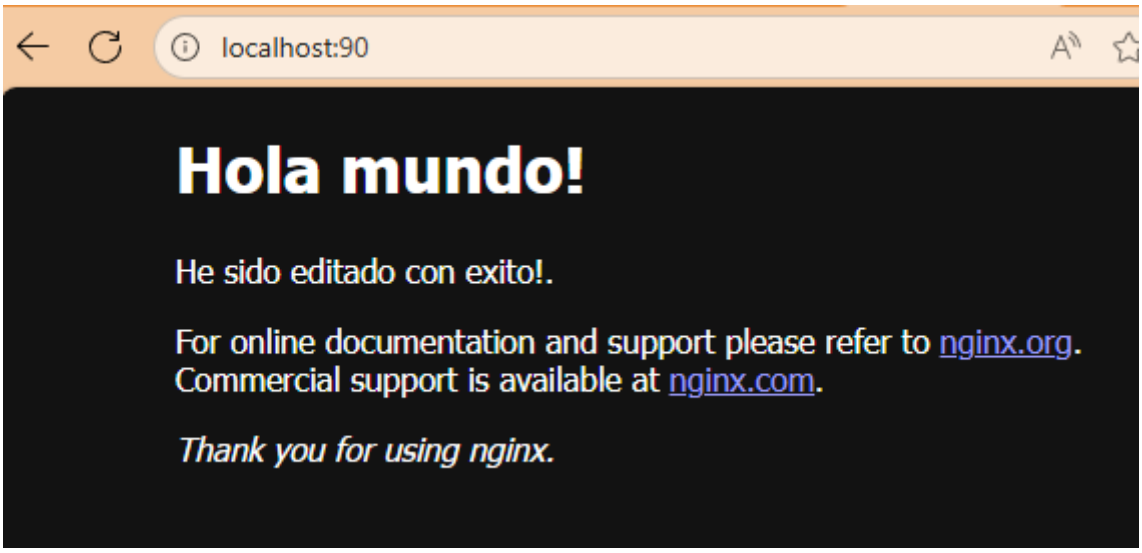


Cada una de esas líneas tienen las siguientes funciones:

- X-Frame-Options: DENY -> Evita que la página se cargue en iframes de otros sitios.
- X-Content-Type-Options: nosniff -> Fuerza al navegador a respetar el tipo de contenido declarado, evitando interpretación maliciosa de archivos.
- X-XSS-Protection: 1; mode=block -> Bloquea la página si se detecta un ataque XSS en navegadores que lo soportan.
- Referrer-Policy: no-referrer -> No envía información de la página de origen, protegiendo la privacidad de la URL.

Salimos y guardamos los cambios, reiniciamos y revisamos que todo siga funcionando

```
nginx -s reload
```



Como se ve, sigue funcionando.

Finalmente, creamos y conectamos tanto “cliente2” como “servidorNginx” a la red que creamos al inicio, la cual es “red_servidor”

```
docker network connect red_servidor cliente2
docker network connect red_servidor servidorNginx
```

- Entramos a “cliente2” para ver si puede ver lo que hay en “servidorNginx”

```
docker start cliente2

docker exec -it cliente2 bash
```

- El ping funciona

```
root@7ccb47b792ec:/# ping servidorNginx
PING servidorNginx (172.25.0.2) 56(84) bytes of data.
64 bytes from servidorNginx.red_servidor (172.25.0.2): icmp_seq=1 ttl=64 time=0.100 ms
64 bytes from servidorNginx.red_servidor (172.25.0.2): icmp_seq=2 ttl=64 time=0.119 ms
64 bytes from servidorNginx.red_servidor (172.25.0.2): icmp_seq=3 ttl=64 time=0.193 ms
64 bytes from servidorNginx.red_servidor (172.25.0.2): icmp_seq=4 ttl=64 time=0.097 ms
64 bytes from servidorNginx.red_servidor (172.25.0.2): icmp_seq=5 ttl=64 time=0.124 ms
^C
--- servidorNginx ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4040ms
rtt min/avg/max/mdev = 0.097/0.126/0.193/0.034 ms
```

Y el curl también

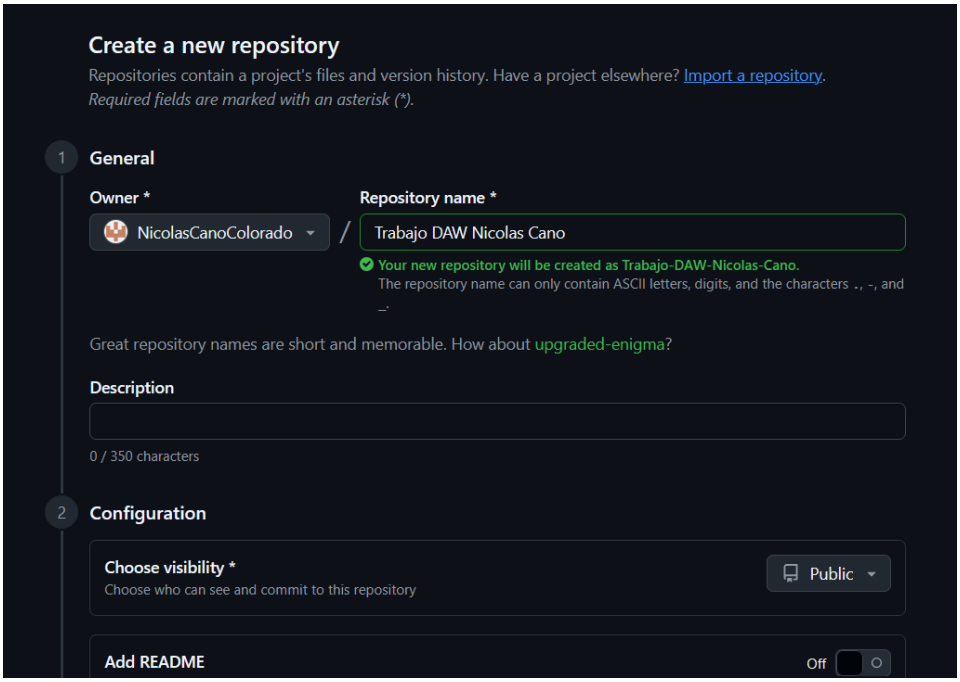
```
root@7ccb47b792ec:/# curl http://servidorNginx:90
<!DOCTYPE html>
<html>
<head>
<title>Soy una aplicacion, y estoy despegando!!!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Hola mundo!</h1>
<p>He sido editado con exito!.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Ademas, se ven los cambios hechos en el index de nuestro servidor.

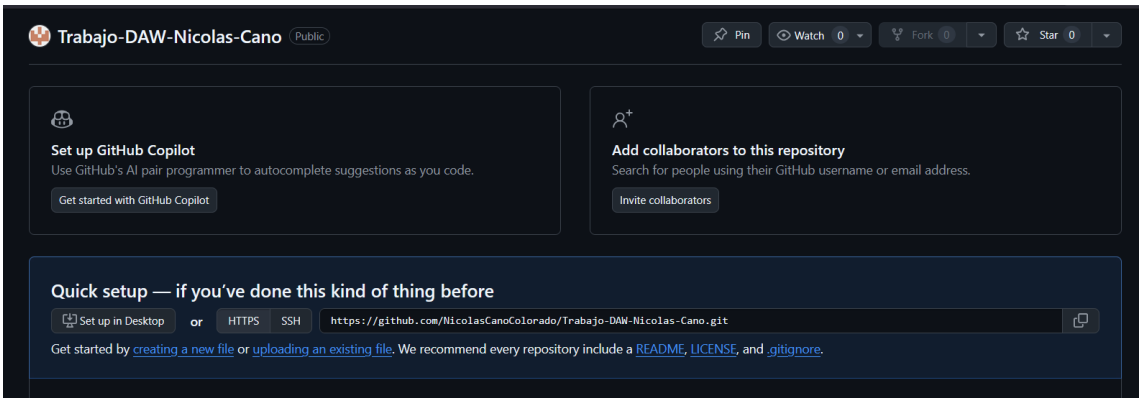
Documentación del despliegue y control de versiones

Creación del repositorio en GitHub

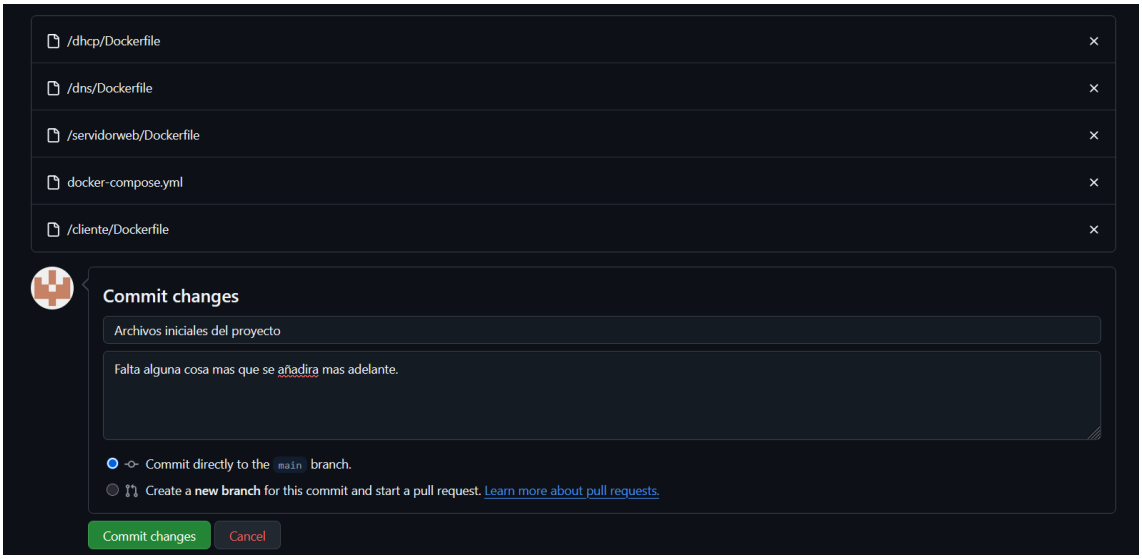
Para guardar el proyecto, se subirá a github. Para ellos se creará el siguiente repositorio:



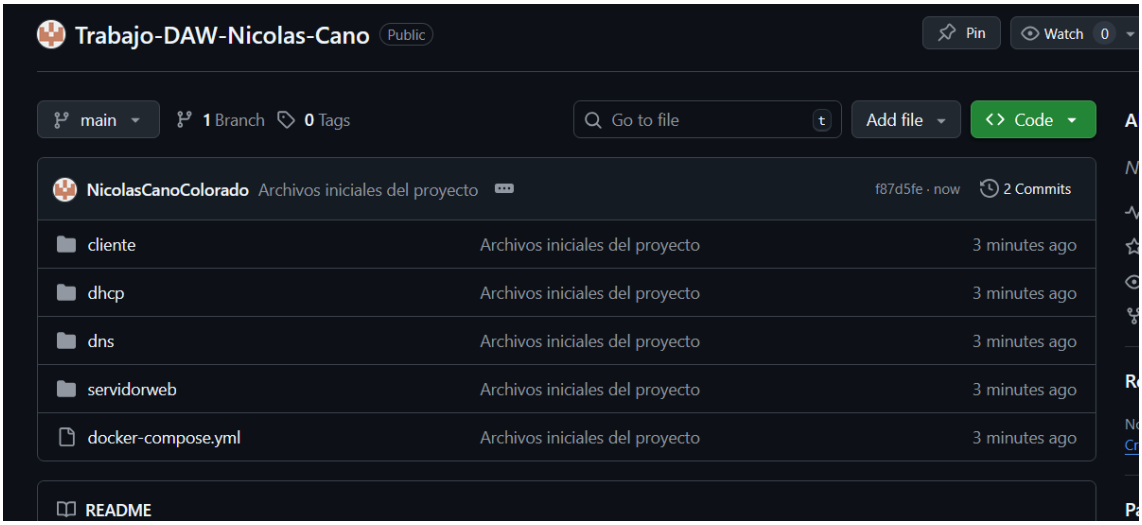
Luego de haber creado el repositorio, se suben los archivos. Para ello, se hace click en “upload existing files”:



Se suben los archivos con una breve descripción:

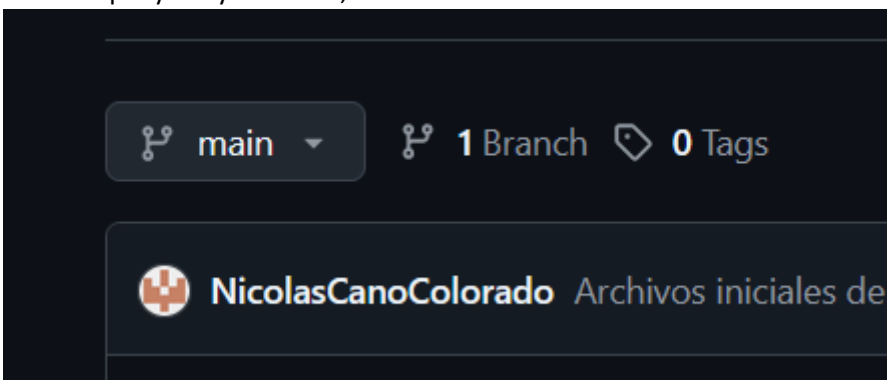


Se hace click en comit y nos queda el repositorio así:

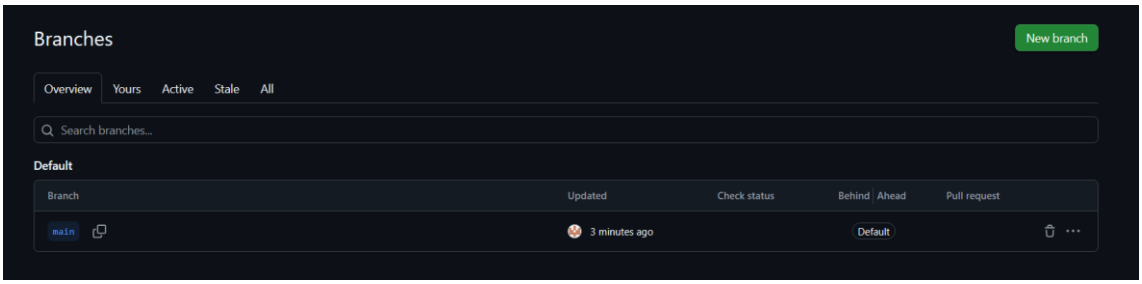


Creación de otra rama

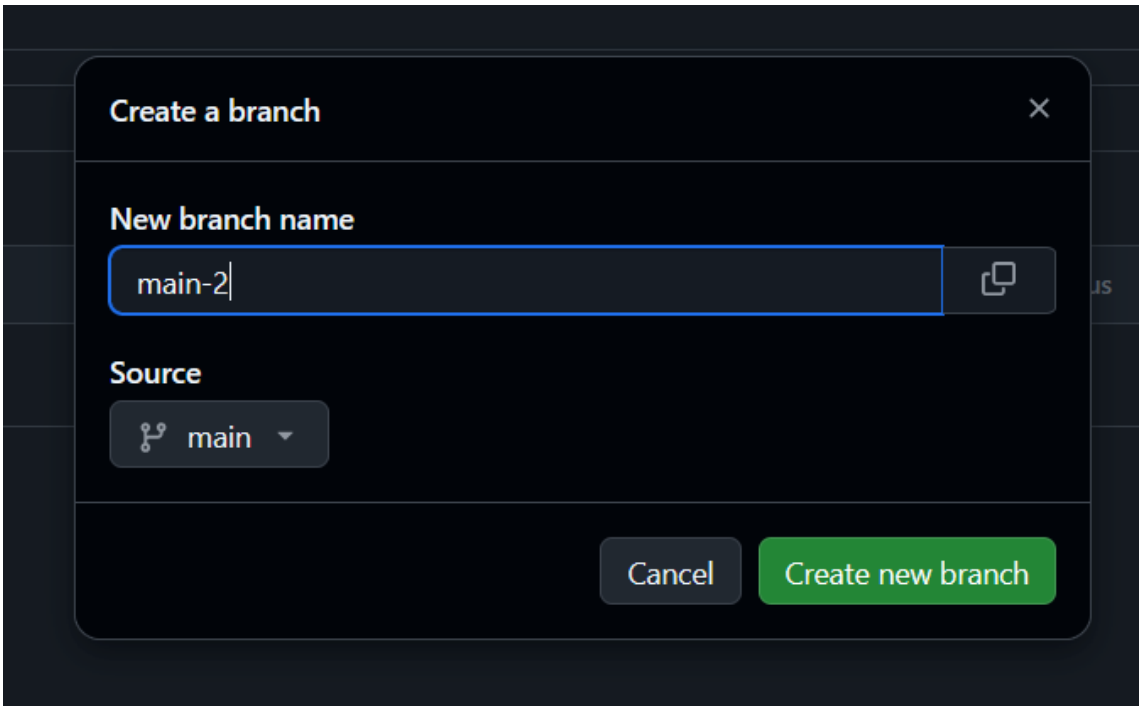
Ya una vez se tiene el repositorio, se crea una rama para añadir archivos faltantes sin tocar lo que ya hay. Primero, se hace click en “branch”



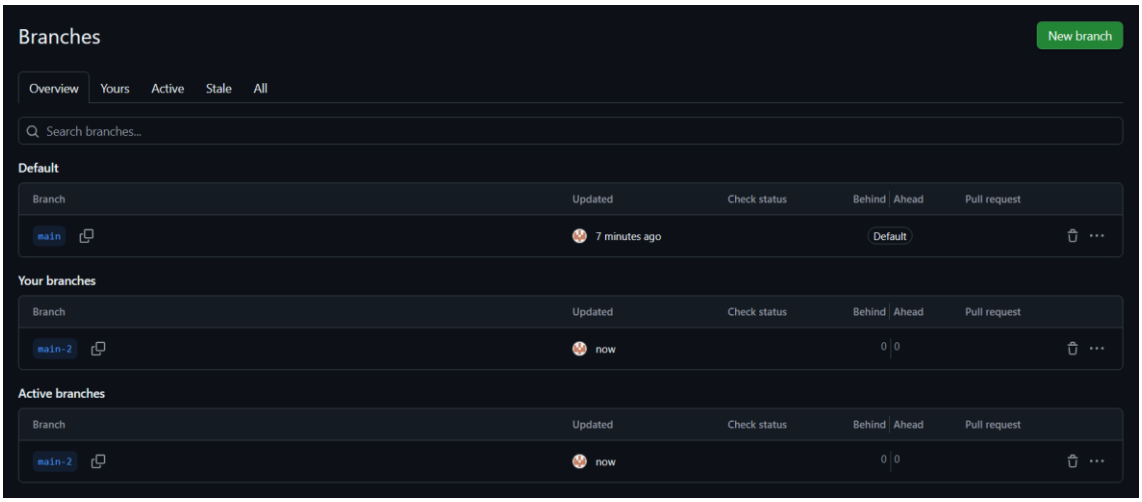
Y en “new branch”:



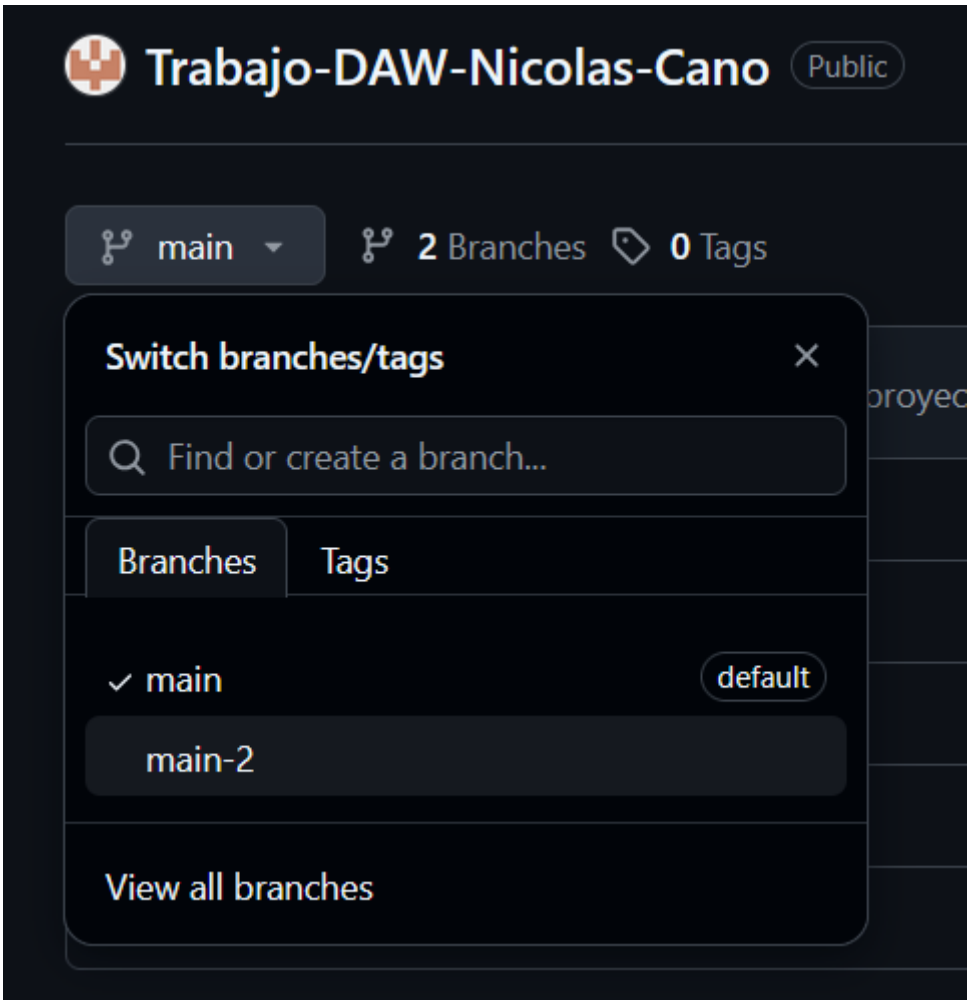
Se llamara “main-2” y se selecciona a partir de cual se va a hacer, o sea, “main”, que es la unica que hay:



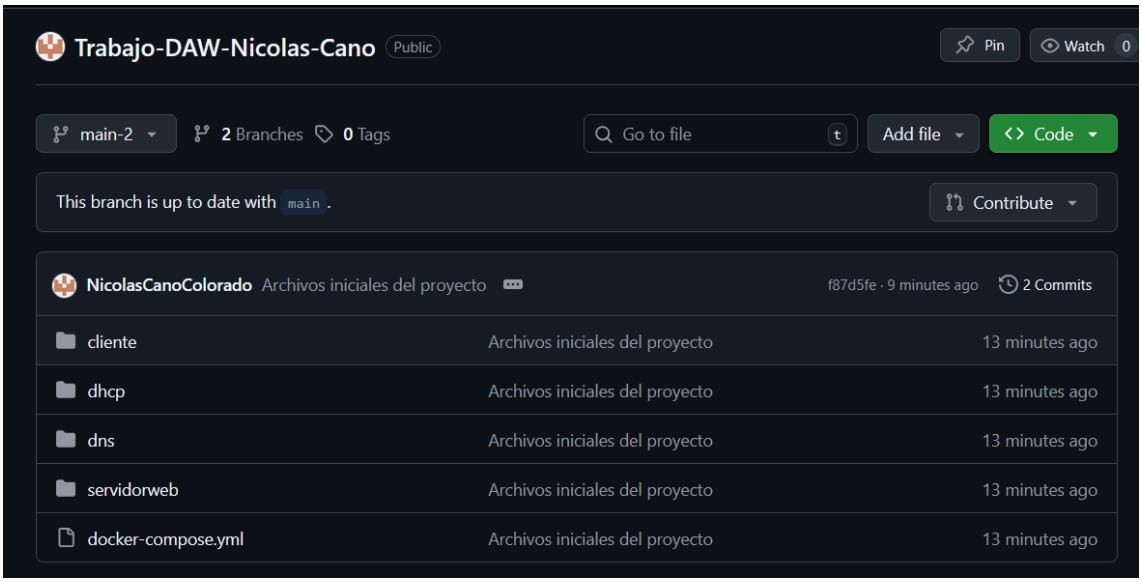
Se le da a “create new branch” para crearla. Despues de eso, se ve como aparece en la lista de branches:



De ahi, se vuelve al repositorio y seleccionamos la branch:

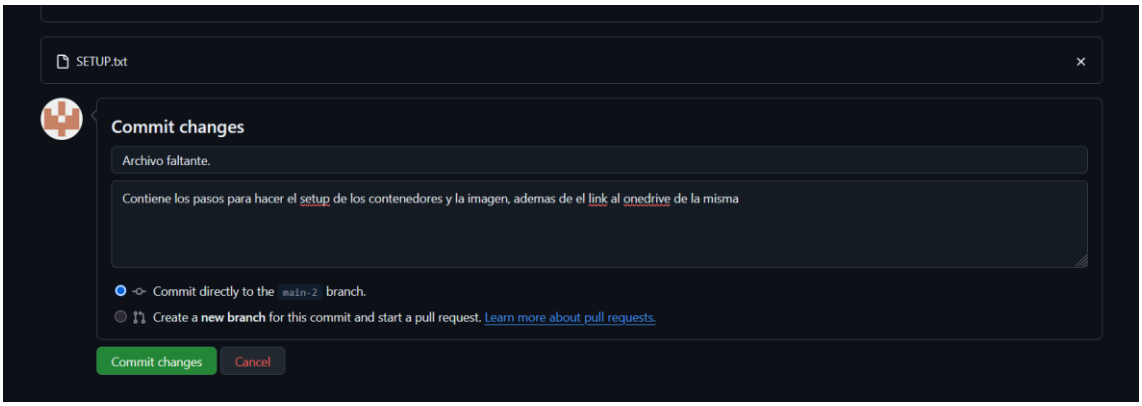


Y listo. Ya la Branch esta hecha. Se puede ver en esa misma esquina donde se veía main, que ahora se ve main-2:

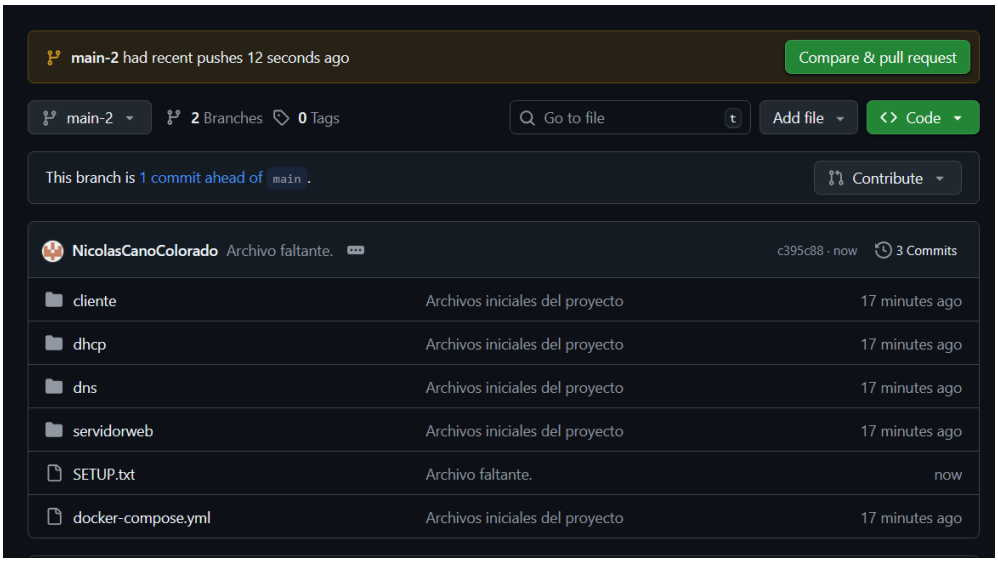


Añadir mas archivos, “merge” y “pull request”

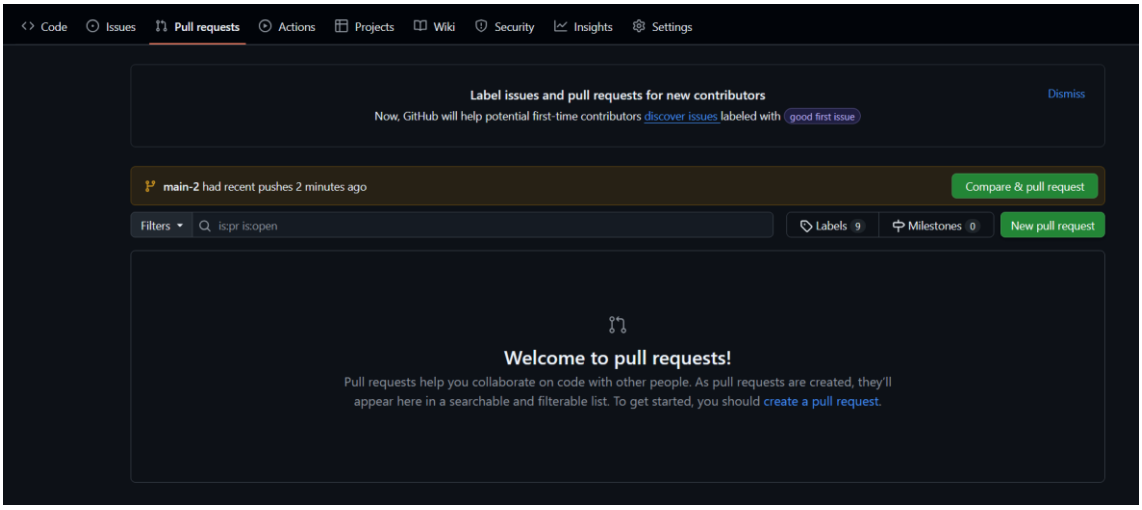
Se hace click en “add file” para añadir un archivo, igual que como se añadieron los primeros archivos:



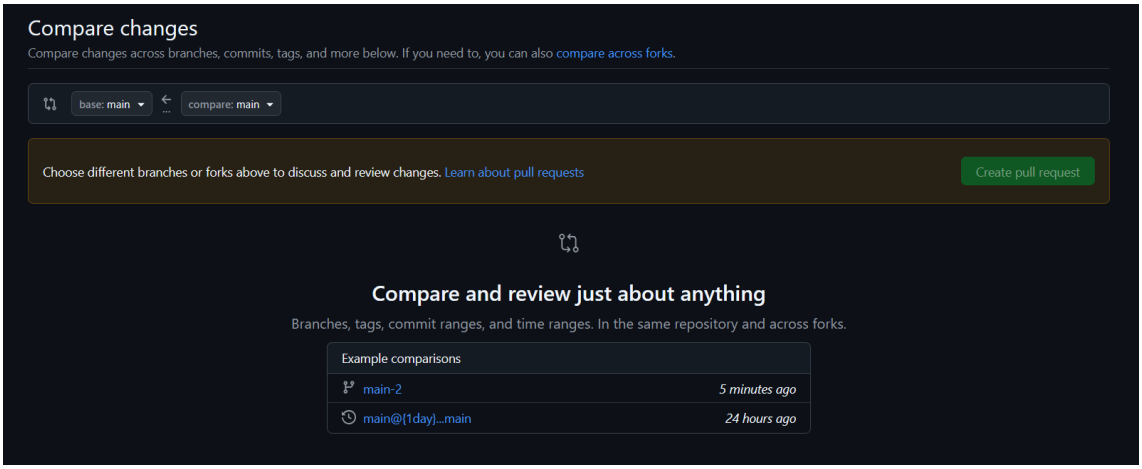
Se hace click en ”commit changes” y listo, nuestra rama “main-2” esta actualizada con el archivo faltante.



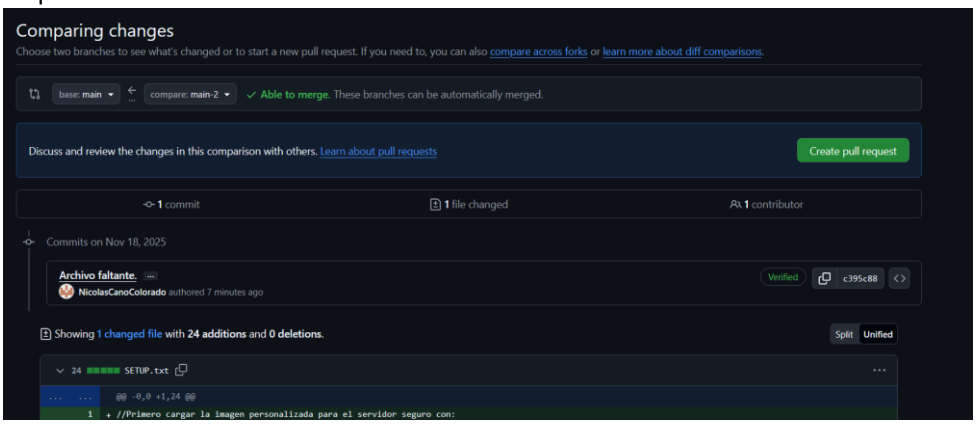
Ahora solo falta hacer “merge” de la rama “main-2” a la rama “main”, lo cual es, básicamente, fusionarlas para que sean una sola de nuevo. Para ello, se va a “pull requests”:



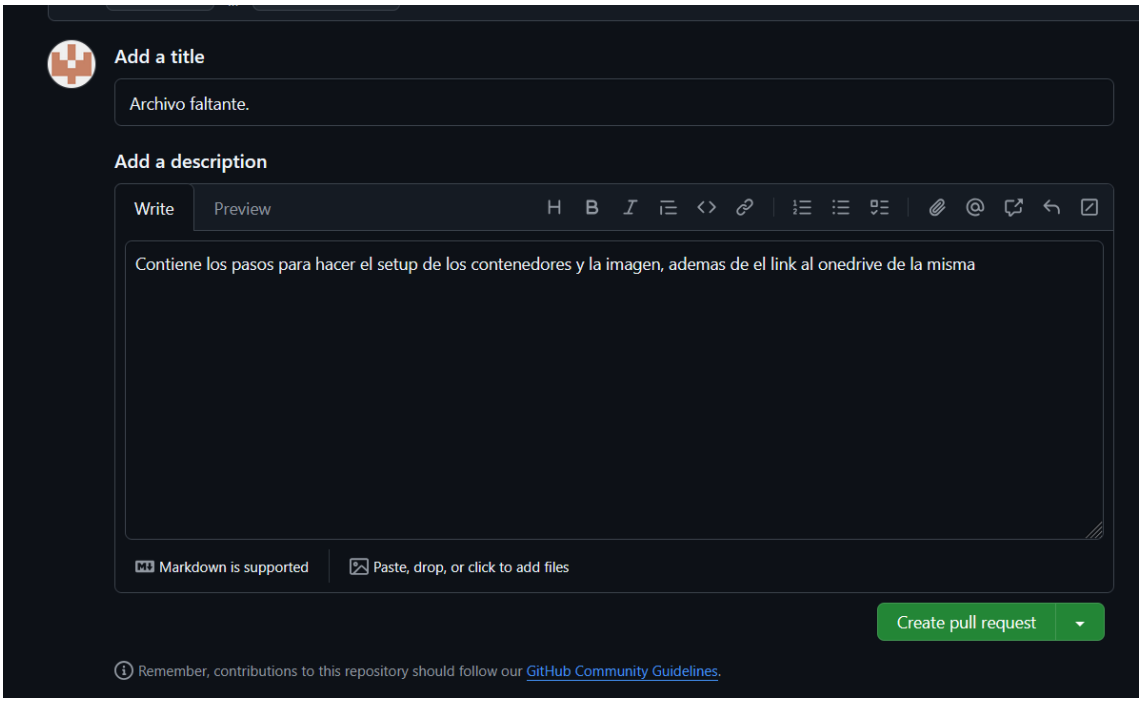
Se le da a “create pull request”. Se muestran las ramas listas para merge, que en este caso solo es una, la cual es “main-2”:



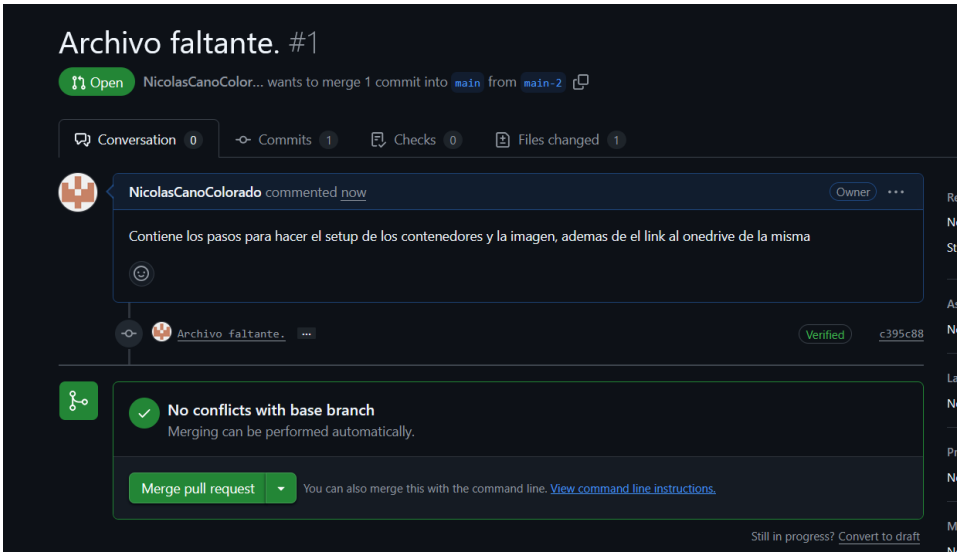
Se le da y se muestra que se ha cambiado/añadido. De ahí, se le da a “create pull request”:

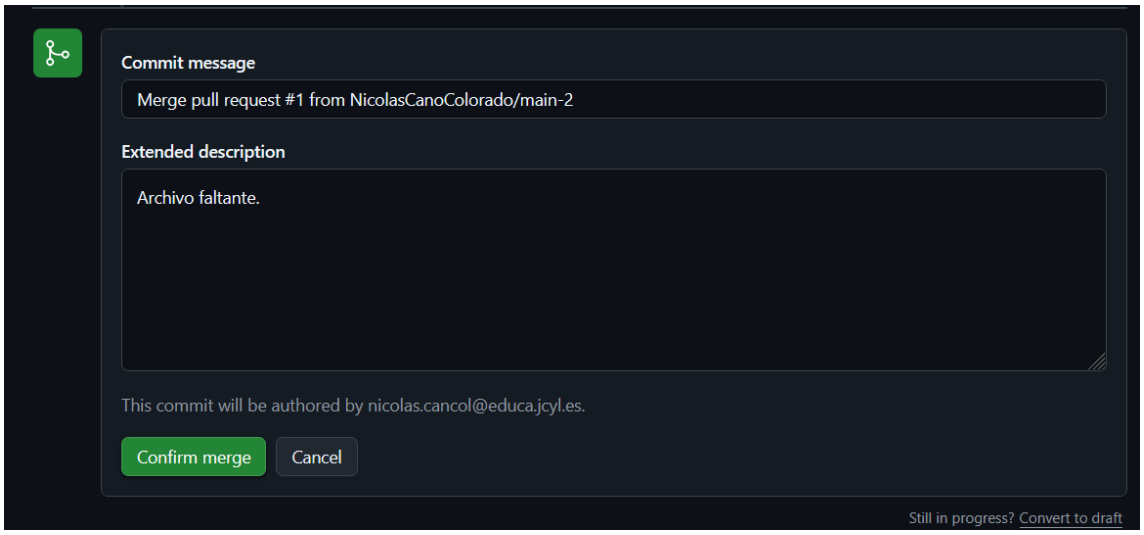


Pide un título y una descripcion. En este caso, se ven lo que se puso al añadir el archivo, así que se deja así y le damos a “create pull request”:

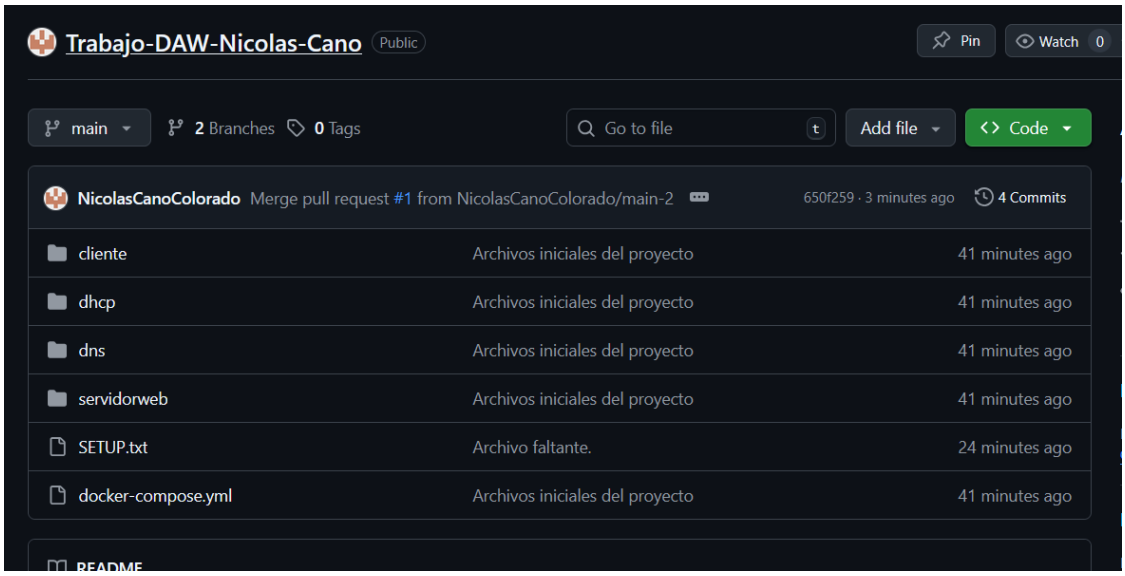


Lo que se acabada de hacer es una petición para que se acepten los cambios. Si fuera el repositorio de otro, habría que esperar a que nos aceptaran tal peticion. En este caso, se acepta la propia peticion haciendo click en “merge pull request” y confirmando con un mensaje opcional:





Y listo. Ya tenemos los cambios hechos originalmente en “main-2” reflejados en “main”:



Con esto, el repositorio ya estaría listo. El enlace al repositorio es:

https://github.com/NicolasCanoColorado/Trabajo-DAW-Nicolas-Cano

Documentación

Sin contar con SETUP.txt, este documento incluye toda la documentación del proyecto y será añadido al repositorio como .docx y como .pdf