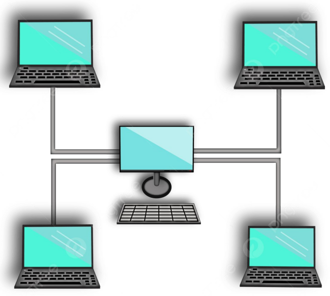


SERVIDORES WEB Y REDES EN DOCKER

docker

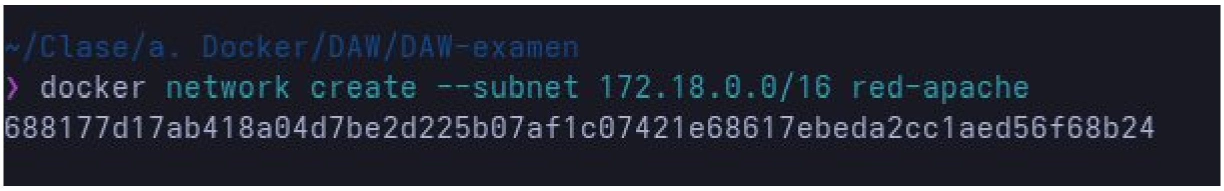
Arquitectura de red

La red estará compuesta por:

- Un adaptador de red Docker de tipo bridge, con la red 172.18.0.0/16, y puerta de enlace 172.18.0.1
- Un contenedor apache con IP 172.18.0.2
- Un contenedor alpine cliente con IP 172.18.0.3

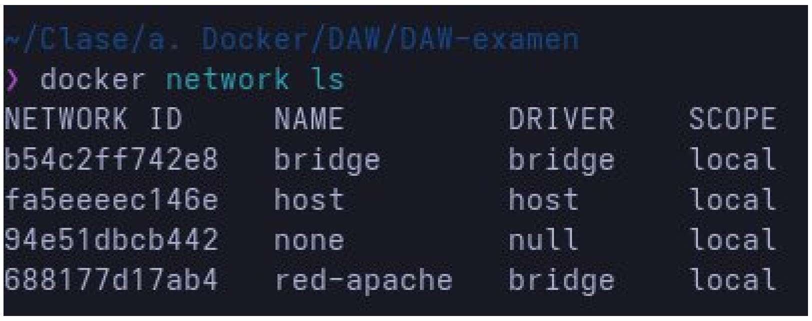
Creación de red

Creamos la red



```
~/Clase/a. Docker/DAW/DAW-examen
❯ docker network create --subnet 172.18.0.0/16 red-apache
688177d17ab418a04d7be2d225b07af1c07421e68617ebeda2cc1aed56f68b24
```

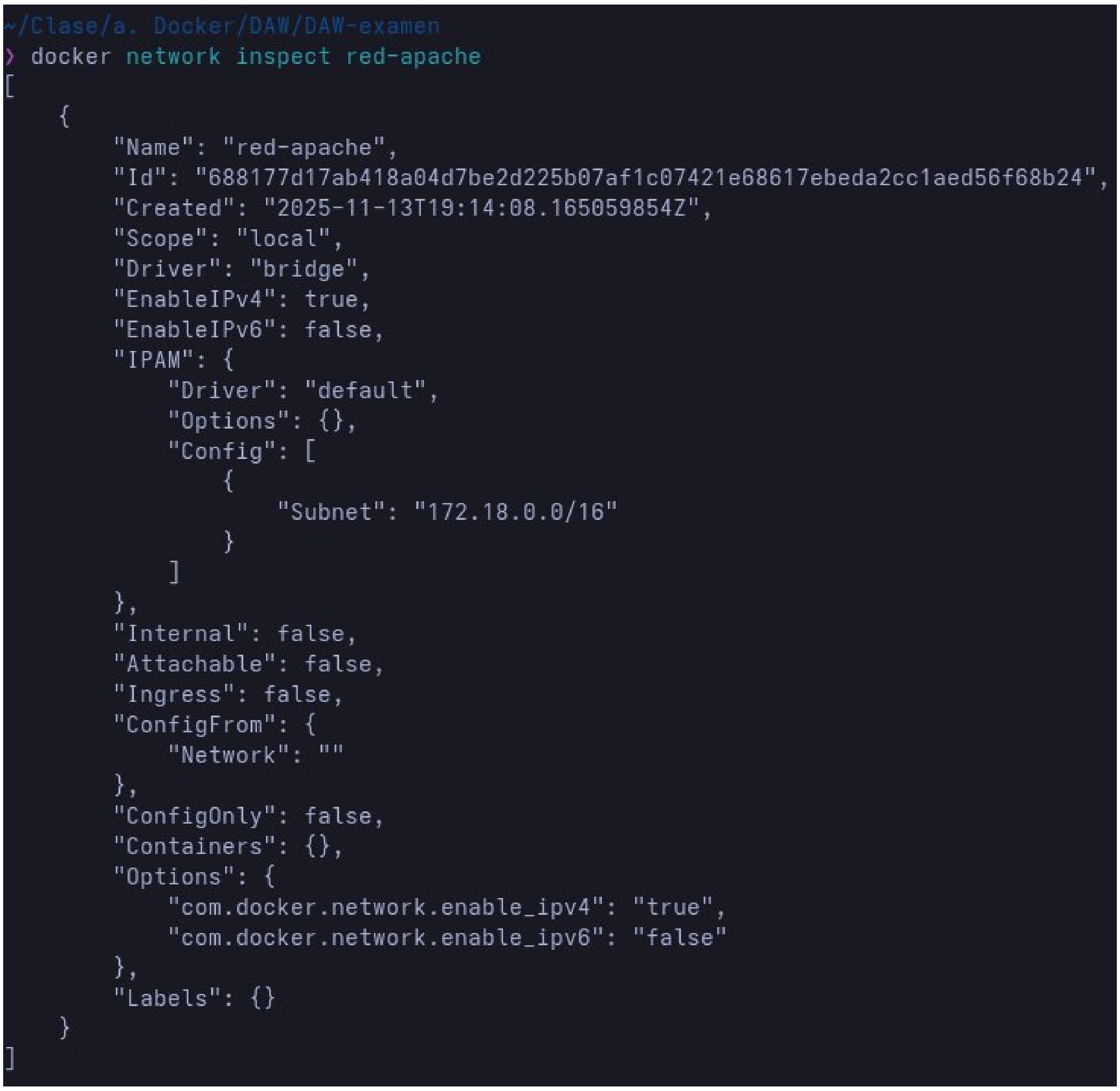
Verificamos que se ha creado bien



```
~/Clase/a. Docker/DAW/DAW-examen
❯ docker network ls
NETWORK ID     NAME         DRIVER    SCOPE
b54c2ff742e8   bridge       bridge    local
fa5eeeec146e   host         host      local
94e51dbcb442   none         null      local
688177d17ab4   red-apache   bridge    local
```

```
~/Clase/a. Docker/DAW/DAW-examen
❯ docker network create --subnet 172.18.0.0/16 red-apache
688177d17ab418a04d7be2d225b07af1c07421e68617ebeda2cc1aed56f68b24
```

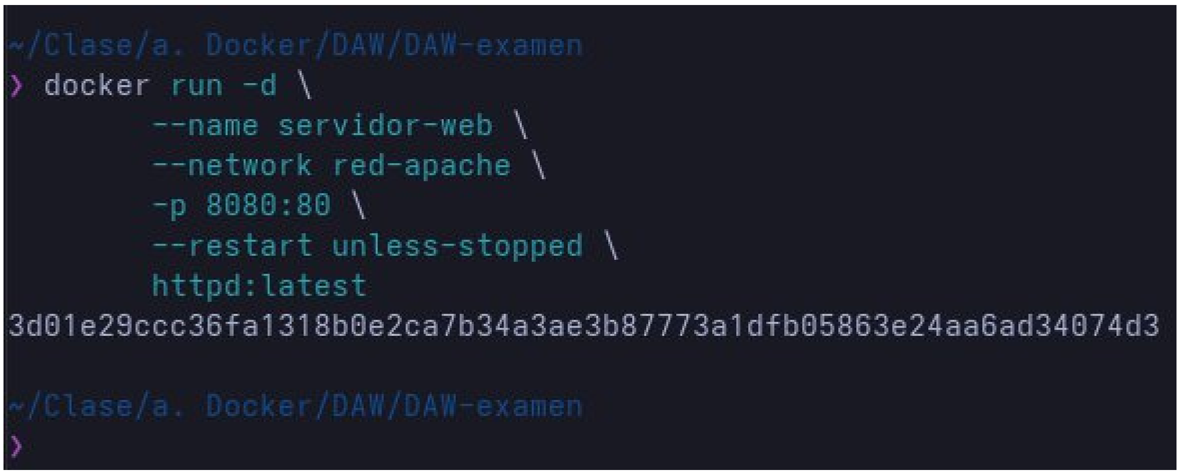
Vemos con inspect si todo es correcto



```
~/Clase/a. Docker/DAW/DAW-examen
❯ docker network inspect red-apache
[
    {
        "Name": "red-apache",
        "Id": "688177d17ab418a04d7be2d225b07af1c07421e68617ebeda2cc1aed56f68b24",
        "Created": "2025-11-13T19:14:08.165059854Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv4": true,
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {
            "com.docker.network.enable_ipv4": "true",
            "com.docker.network.enable_ipv6": "false"
        },
        "Labels": {}
    }
]
```

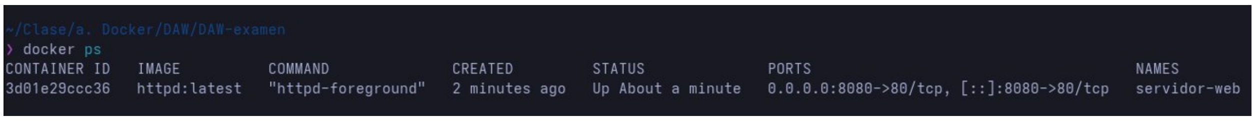
Despliegue del servidor

Creamos un contenedor docker con nuestro servidor apache, le conectamos a nuestra red bridge, y mapeamos el puerto 80 del contenedor al puerto 8080 del anfitrión



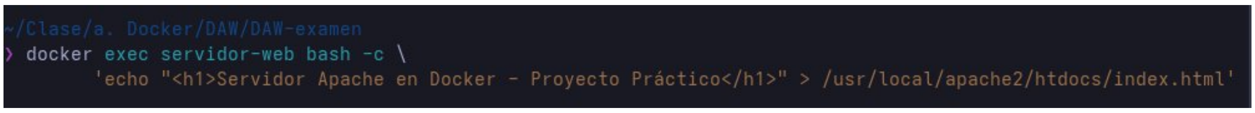
```
~/Clase/a. Docker/DAW/DAW-examen
❯ docker run -d \
        --name servidor-web \
        --network red-apache \
        -p 8080:80 \
        --restart unless-stopped \
        httpd:latest
3d01e29ccc36fa1318b0e2ca7b34a3ae3b87773a1dfb05863e24aa6ad34074d3

~/Clase/a. Docker/DAW/DAW-examen
❯
```

Verificamos que se está ejecutando



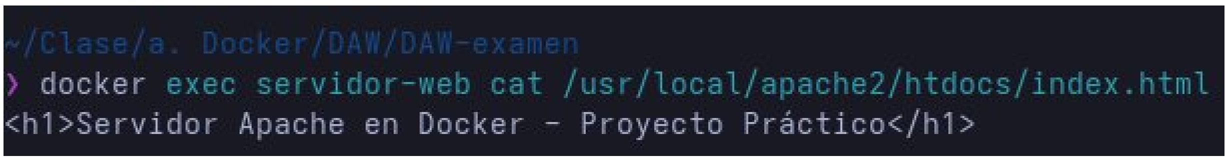
```
~/Clase/a. Docker/DAW/DAW-examen
❯ docker ps
CONTAINER ID   IMAGE          COMMAND              CREATED         STATUS           PORTS                                     NAMES
3d01e29ccc36   httpd:latest   "httpd-foreground"   2 minutes ago   Up About a minute   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   servidor-web
```

Creamos un pequeño index



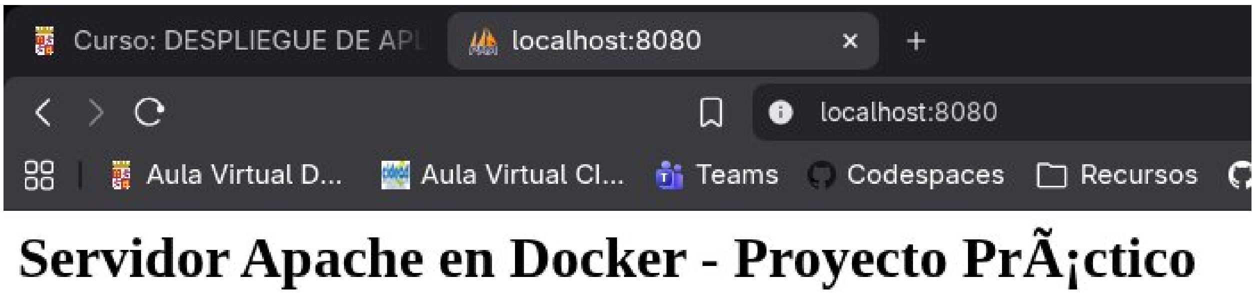
```
~/Clase/a. Docker/DAW/DAW-examen
❯ docker exec servidor-web bash -c \
        'echo "<h1>Servidor Apache en Docker - Proyecto Práctico</h1>" > /usr/local/apache2/htdocs/index.html'
```

Verificamos que nuestro index está como queremos



```
~/Clase/a. Docker/DAW/DAW-examen
❯ docker exec servidor-web cat /usr/local/apache2/htdocs/index.html
<h1>Servidor Apache en Docker - Proyecto Práctico</h1>
```

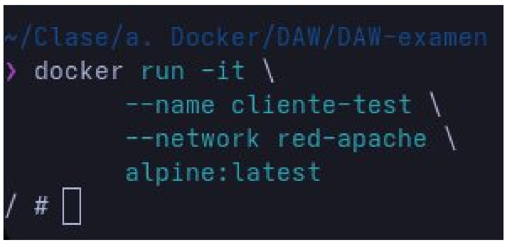
Probamos desde la máquina anfitrión con localhost:8080



Servidor Apache en Docker - Proyecto PrÃ¡ctico

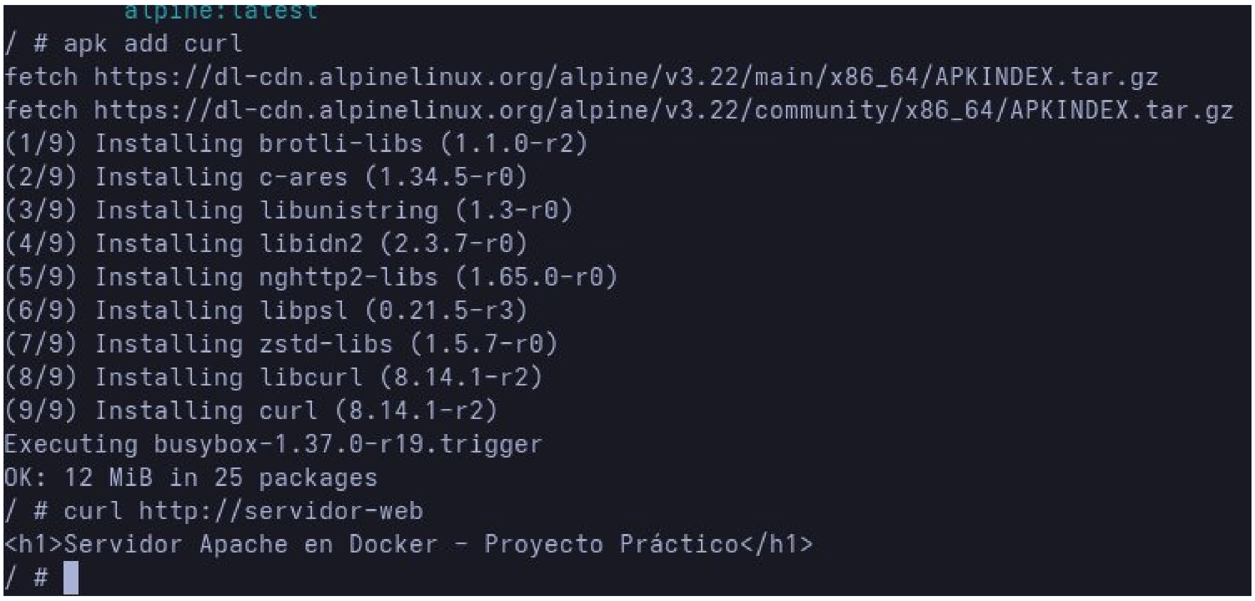
Despliegue del cliente

Creamos un cliente alpine y lo conectamos a nuestra red:



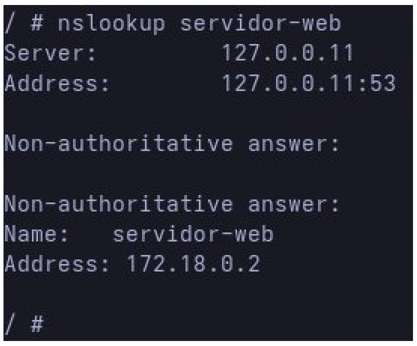
```
~/Clase/a. Docker/DAW/DAW-examen
❯ docker run -it \
        --name cliente-test \
        --network red-apache \
        alpine:latest
/ # 
```

Dentro del contenedor probamos la conectividad con curl



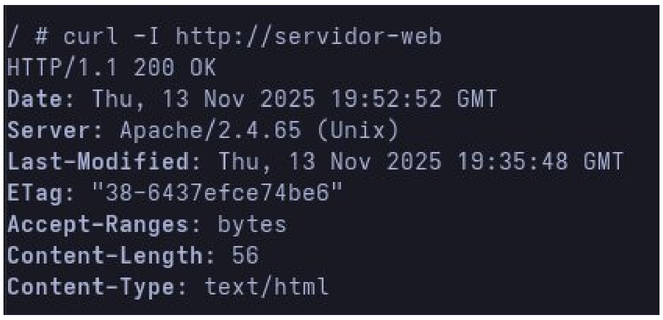
```
        alpine:latest
/ # apk add curl
fetch https://dl-cdn.alpinelinux.org/alpine/v3.22/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.22/community/x86_64/APKINDEX.tar.gz
(1/9) Installing brotli-libs (1.1.0-r2)
(2/9) Installing c-ares (1.34.5-r0)
(3/9) Installing libunistring (1.3-r0)
(4/9) Installing libidn2 (2.3.7-r0)
(5/9) Installing nghttp2-libs (1.65.0-r0)
(6/9) Installing libpsl (0.21.5-r3)
(7/9) Installing zstd-libs (1.5.7-r0)
(8/9) Installing libcurl (8.14.1-r2)
(9/9) Installing curl (8.14.1-r2)
Executing busybox-1.37.0-r19.trigger
OK: 12 MiB in 25 packages
/ # curl http://servidor-web
<h1>Servidor Apache en Docker - Proyecto Práctico</h1>
/ # 
```

Probamos si el DNS interno de docker funciona



```
/ # nslookup servidor-web
Server:         127.0.0.11
Address:        127.0.0.11:53

Non-authoritative answer:

Non-authoritative answer:
Name:   servidor-web
Address: 172.18.0.2

/ #
```

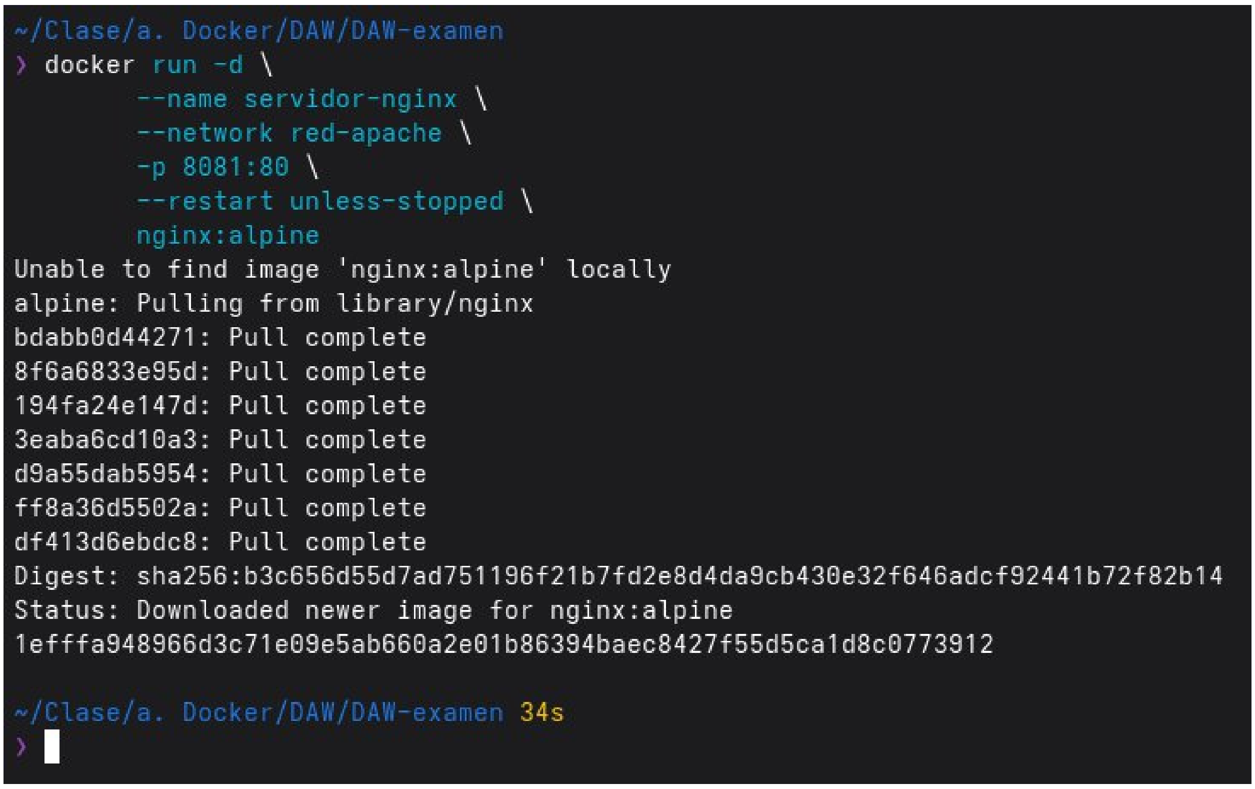
Probamos la conectividad por el puerto 80 de un contenedor a otro



```
/ # curl -I http://servidor-web
HTTP/1.1 200 OK
Date: Thu, 13 Nov 2025 19:52:52 GMT
Server: Apache/2.4.65 (Unix)
Last-Modified: Thu, 13 Nov 2025 19:35:48 GMT
ETag: "38-6437efce74be6"
Accept-Ranges: bytes
Content-Length: 56
Content-Type: text/html
```

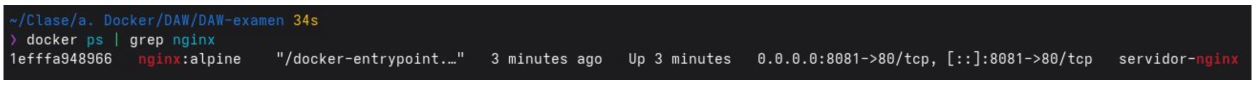
Funcionalidad extra

Añadimos un contenedor de NGINX. Este puede funcionar como proxy inverso o servidor web alternativo.



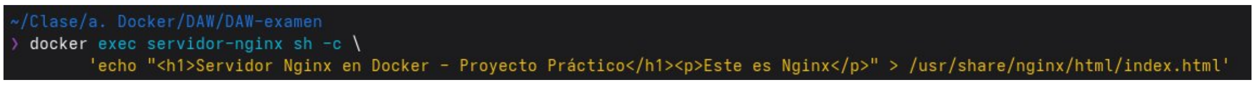
```
~/Clase/a. Docker/DAW/DAW-examen
❯ docker run -d \
        --name servidor-nginx \
        --network red-apache \
        -p 8081:80 \
        --restart unless-stopped \
        nginx:alpine
Unable to find image 'nginx:alpine' locally
alpine: Pulling from library/nginx
bdabb0d44271: Pull complete
8f6a6833e95d: Pull complete
194fa24e147d: Pull complete
3eaba6cd10a3: Pull complete
d9a55dab5954: Pull complete
ff8a36d5502a: Pull complete
df413d6ebdc8: Pull complete
Digest: sha256:b3c656d55d7ad751196f21b7fd2e8d4da9cb430e32f646adcf92441b72f82b14
Status: Downloaded newer image for nginx:alpine
1efffa948966d3c71e09e5ab660a2e01b86394baec8427f55d5ca1d8c0773912

~/Clase/a. Docker/DAW/DAW-examen 34s
❯
```

Comprobamos que está ejecutándose con un grep



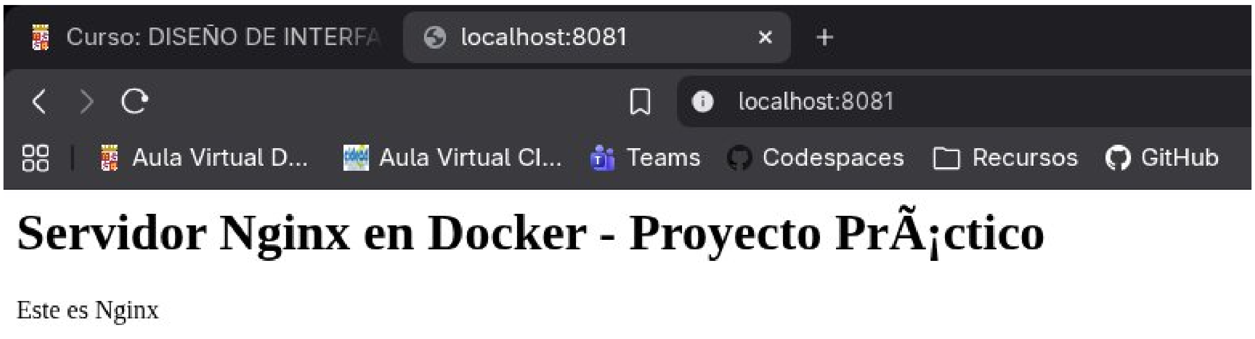
```
~/Clase/a. Docker/DAW/DAW-examen 34s
❯ docker ps | grep nginx
1efffa948966   nginx:alpine   "/docker-entrypoint.…"   3 minutes ago   Up 3 minutes   0.0.0.0:8081->80/tcp, [::]:8081->80/tcp   servidor-nginx
```

Metemos en el index un poco de código para comprobar que funcione



```
~/Clase/a. Docker/DAW/DAW-examen
❯ docker exec servidor-nginx sh -c \
        'echo "<h1>Servidor Nginx en Docker - Proyecto Práctico</h1><p>Este es Nginx</p>" > /usr/share/nginx/html/index.html'
```

Probamos desde un navegador que funcione

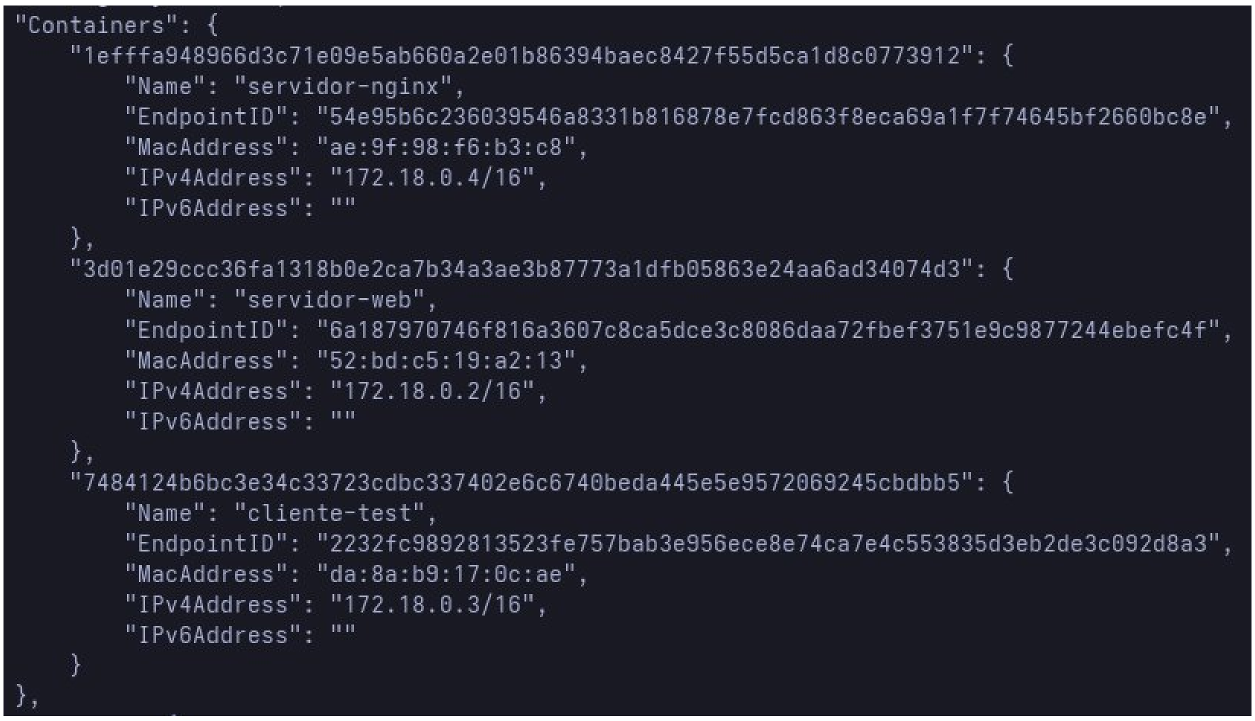


Servidor Nginx en Docker - Proyecto PrÃ¡ctico

Este es Nginx

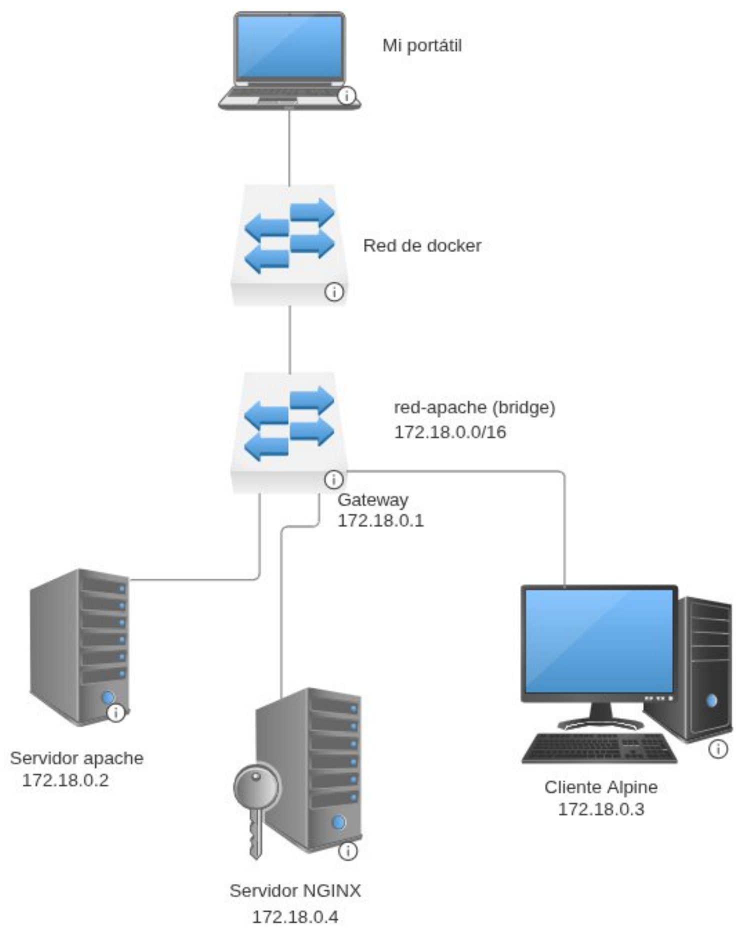
Arquitectura de red final

Con los contenedores ya realizados, vemos cómo nos ha quedado la arquitectura de red, esto nos muestra un network inspect:



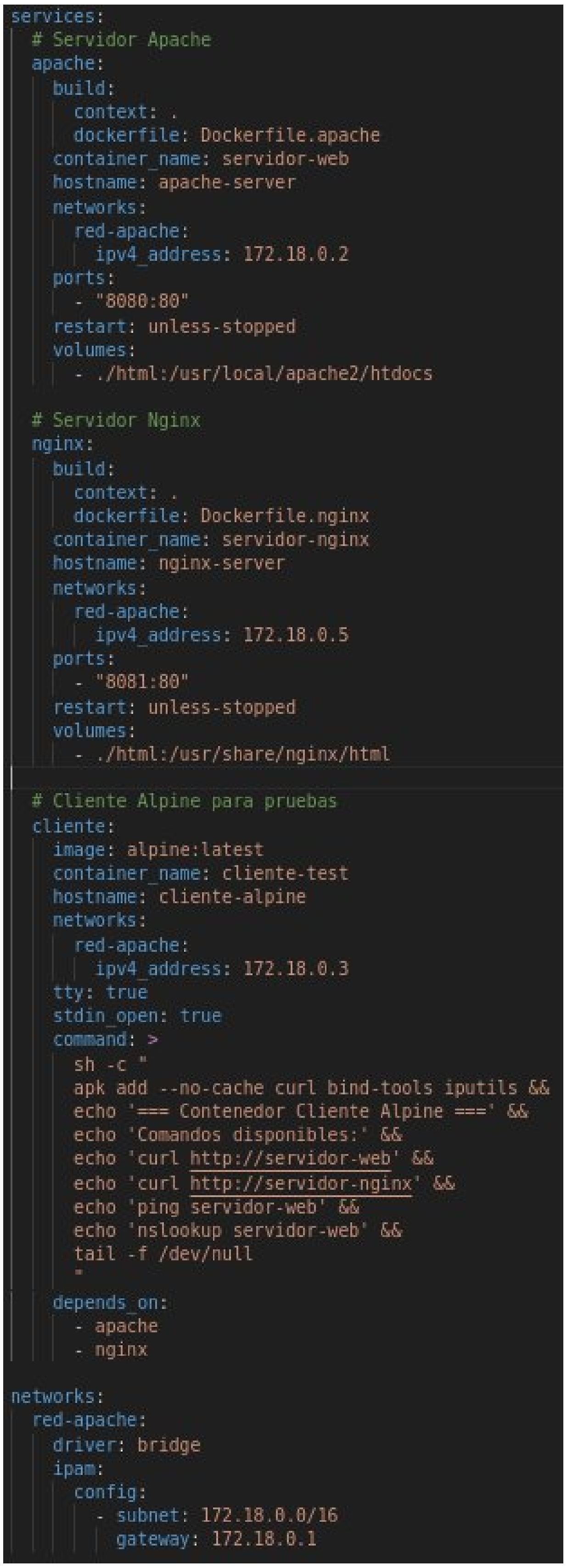
```
"Containers": {
    "1efffa948966d3c71e09e5ab660a2e01b86394baec8427f55d5ca1d8c0773912": {
        "Name": "servidor-nginx",
        "EndpointID": "54e95b6c236039546a8331b816878e7fcd863f8eca69a1f7f74645bf2660bc8e",
        "MacAddress": "ae:9f:98:f6:b3:c8",
        "IPv4Address": "172.18.0.4/16",
        "IPv6Address": ""
    },
    "3d01e29ccc36fa1318b0e2ca7b34a3ae3b87773a1dfb05863e24aa6ad34074d3": {
        "Name": "servidor-web",
        "EndpointID": "6a187970746f816a3607c8ca5dce3c8086daa72fbef3751e9c9877244ebefc4f",
        "MacAddress": "52:bd:c5:19:a2:13",
        "IPv4Address": "172.18.0.2/16",
        "IPv6Address": ""
    },
    "7484124b6bc3e34c33723cdbc337402e6c6740beda445e5e9572069245cbdbb5": {
        "Name": "cliente-test",
        "EndpointID": "2232fc9892813523fe757bab3e956ece8e74ca7e4c553835d3eb2de3c092d8a3",
        "MacAddress": "da:8a:b9:17:0c:ae",
        "IPv4Address": "172.18.0.3/16",
        "IPv6Address": ""
    }
},
```

Por lo que nos queda esta arquitectura de red:



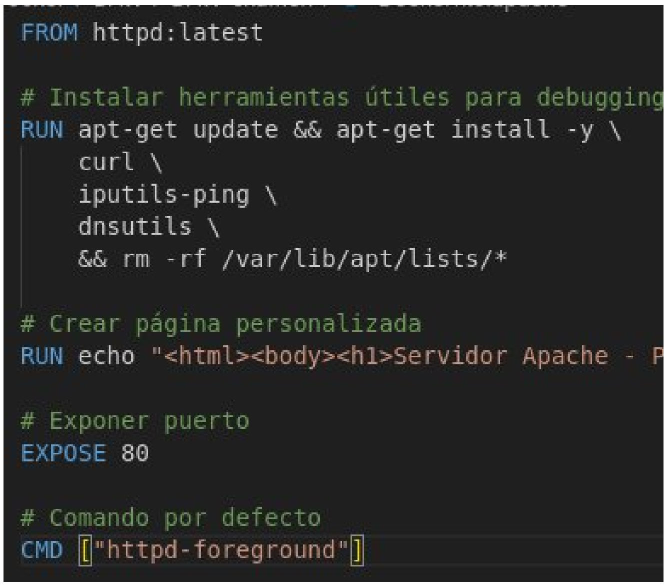
Creación de dockerfile y docker-compose.yml

Una vez hemos creado nuestros contenedores, podemos crear un Dockerfile y un archivo docker-compose.yml para automatizar el proceso de despliegue. Este es el docker-compose.yml que he hecho:



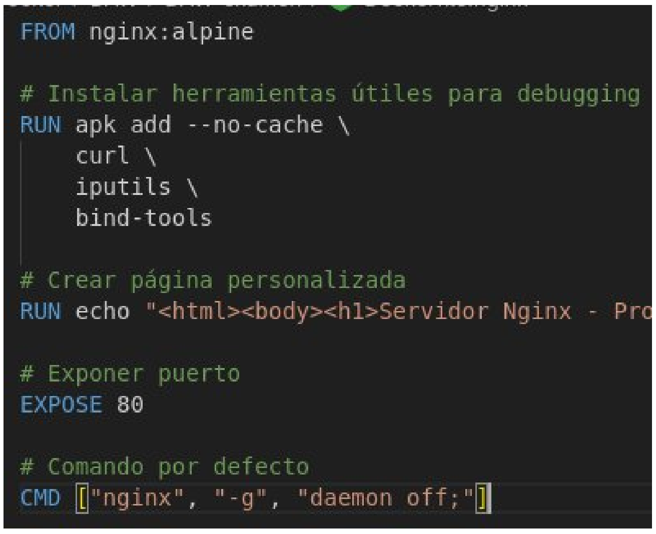
```yaml
services:
  # Servidor Apache
  apache:
    build:
      context: .
      dockerfile: Dockerfile.apache
    container_name: servidor-web
    hostname: apache-server
    networks:
      red-apache:
        ipv4_address: 172.18.0.2
    ports:
      - "8080:80"
    restart: unless-stopped
    volumes:
      - ./html:/usr/local/apache2/htdocs

  # Servidor Nginx
  nginx:
    build:
      context: .
      dockerfile: Dockerfile.nginx
    container_name: servidor-nginx
    hostname: nginx-server
    networks:
      red-apache:
        ipv4_address: 172.18.0.5
    ports:
      - "8081:80"
    restart: unless-stopped
    volumes:
      - ./html:/usr/share/nginx/html

  # Cliente Alpine para pruebas
  cliente:
    image: alpine:latest
    container_name: cliente-test
    hostname: cliente-alpine
    networks:
      red-apache:
        ipv4_address: 172.18.0.3
    tty: true
    stdin_open: true
    command: >
      sh -c "
      apk add --no-cache curl bind-tools iputils &&
      echo '=== Contenedor Cliente Alpine ===' &&
      echo 'Comandos disponibles:' &&
      echo 'curl http://servidor-web' &&
      echo 'curl http://servidor-nginx' &&
      echo 'ping servidor-web' &&
      echo 'nslookup servidor-web' &&
      tail -f /dev/null
      "
    depends_on:
      - apache
      - nginx

networks:
  red-apache:
    driver: bridge
    ipam:
      config:
        - subnet: 172.18.0.0/16
          gateway: 172.18.0.1
```

Este es el dockerfile de apache:



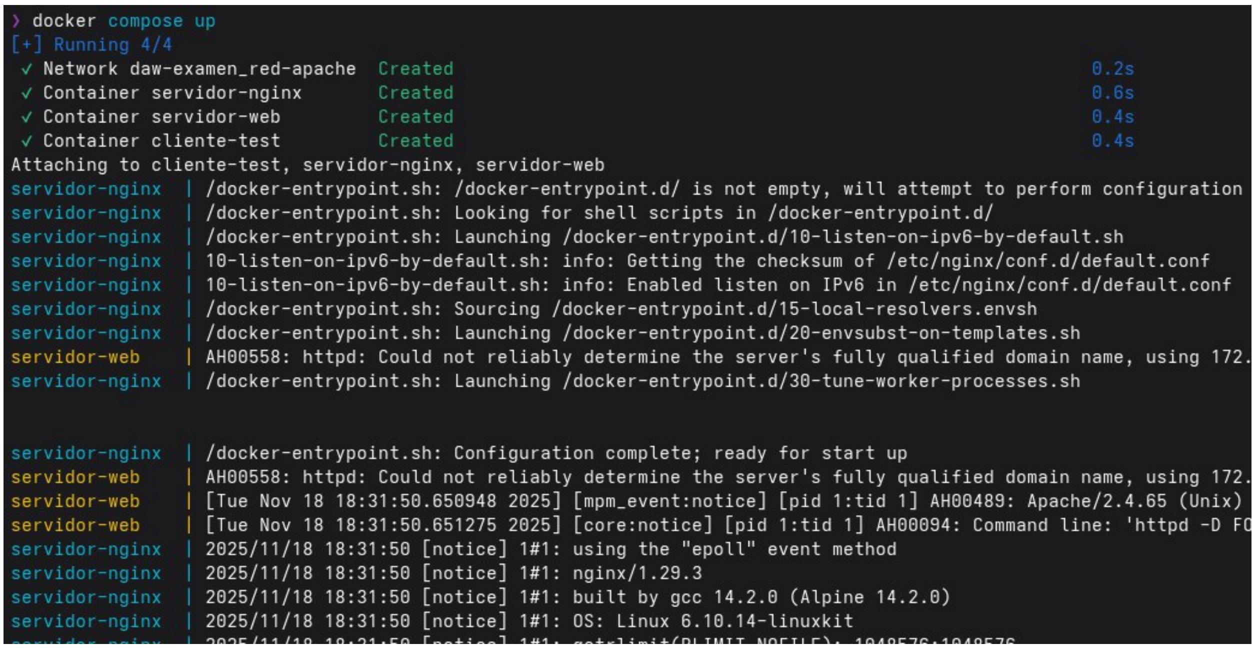
```
FROM httpd:latest

# Instalar herramientas útiles para debugging
RUN apt-get update && apt-get install -y \
    curl \
    iputils-ping \
    dnsutils \
    && rm -rf /var/lib/apt/lists/*

# Crear página personalizada
RUN echo "<html><body><h1>Servidor Apache - P

# Exponer puerto
EXPOSE 80

# Comando por defecto
CMD ["httpd-foreground"]
```

Y este es el dockerfile de nginx:



```
FROM nginx:alpine

# Instalar herramientas útiles para debugging
RUN apk add --no-cache \
    curl \
    iputils \
    bind-tools

# Crear página personalizada
RUN echo "<html><body><h1>Servidor Nginx - Pro

# Exponer puerto
EXPOSE 80

# Comando por defecto
CMD ["nginx", "-g", "daemon off;"]
```

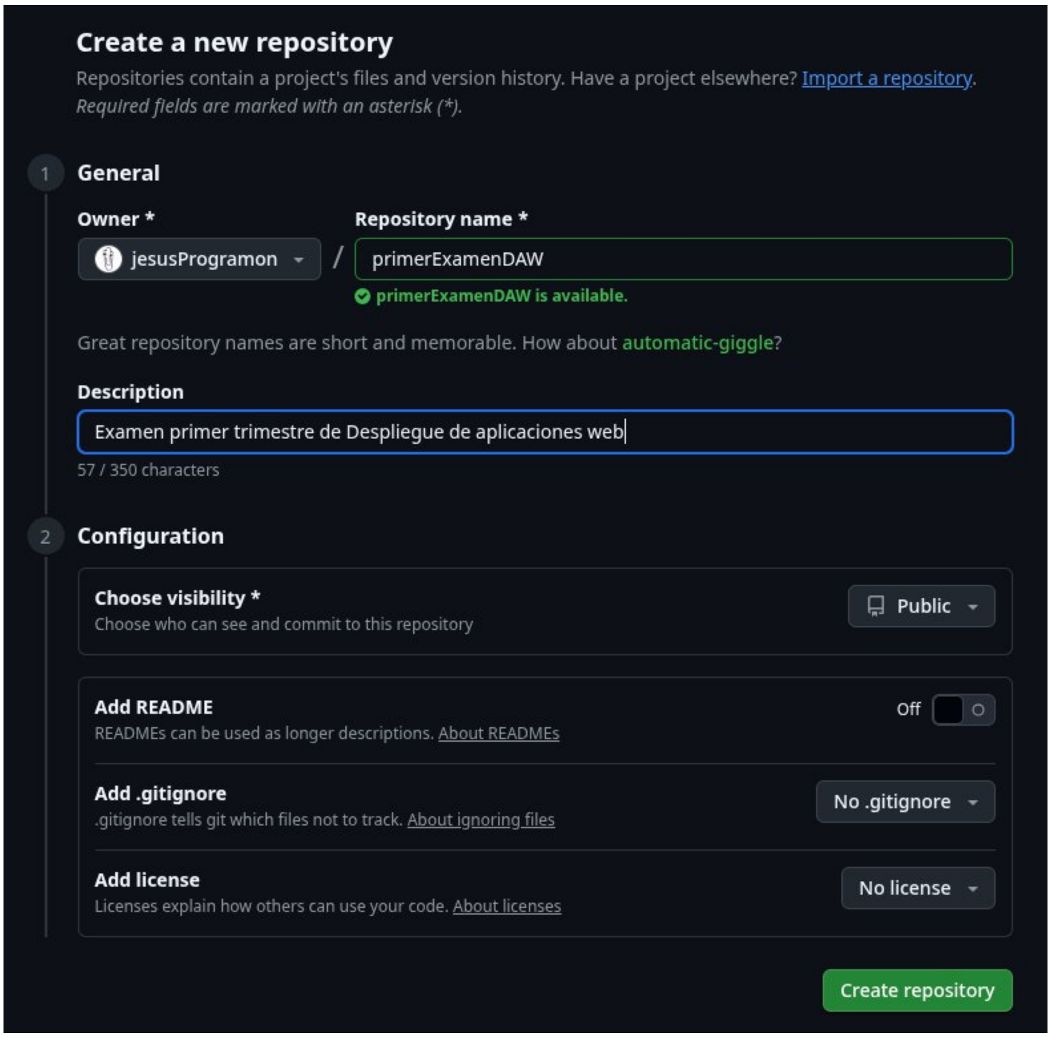
Hacemos un docker compose up y a funcionar



```
❯ docker compose up
[+] Running 4/4
 ✓ Network daw-examen_red-apache  Created                                                          0.2s
 ✓ Container servidor-nginx       Created                                                          0.6s
 ✓ Container servidor-web         Created                                                          0.4s
 ✓ Container cliente-test         Created                                                          0.4s
Attaching to cliente-test, servidor-nginx, servidor-web
servidor-nginx  | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
servidor-nginx  | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
servidor-nginx  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
servidor-nginx  | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
servidor-nginx  | 10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
servidor-nginx  | /docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
servidor-nginx  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
servidor-web    | AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.
servidor-nginx  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh


servidor-nginx  | /docker-entrypoint.sh: Configuration complete; ready for start up
servidor-web    | AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.
servidor-web    | [Tue Nov 18 18:31:50.650948 2025] [mpm_event:notice] [pid 1:tid 1] AH00489: Apache/2.4.65 (Unix)
servidor-web    | [Tue Nov 18 18:31:50.651275 2025] [core:notice] [pid 1:tid 1] AH00094: Command line: 'httpd -D FO
servidor-nginx  | 2025/11/18 18:31:50 [notice] 1#1: using the "epoll" event method
servidor-nginx  | 2025/11/18 18:31:50 [notice] 1#1: nginx/1.29.3
servidor-nginx  | 2025/11/18 18:31:50 [notice] 1#1: built by gcc 14.2.0 (Alpine 14.2.0)
servidor-nginx  | 2025/11/18 18:31:50 [notice] 1#1: OS: Linux 6.10.14-linuxkit
servidor-nginx  | 2025/11/18 18:31:50 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
```

Subir a Github

Primero creamos el repositorio



Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? Import a repository.
Required fields are marked with an asterisk (*).

1 **General**

Owner * jesusProgramon / **Repository name *** primerExamenDAW

primerExamenDAW is available.

Great repository names are short and memorable. How about automatic-giggle?

Description

Examen primer trimestre de Despliegue de aplicaciones web

57 / 350 characters

2 **Configuration**

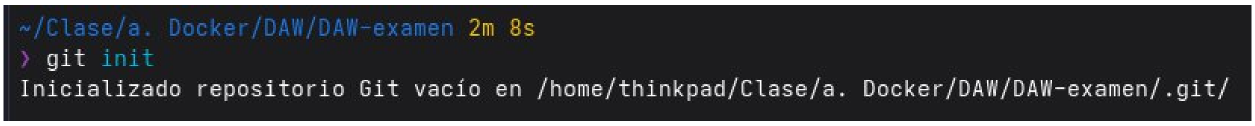
Choose visibility * — Public
Choose who can see and commit to this repository

Add README — Off
READMEs can be used as longer descriptions. About READMEs

Add .gitignore — No .gitignore
.gitignore tells git which files not to track. About ignoring files

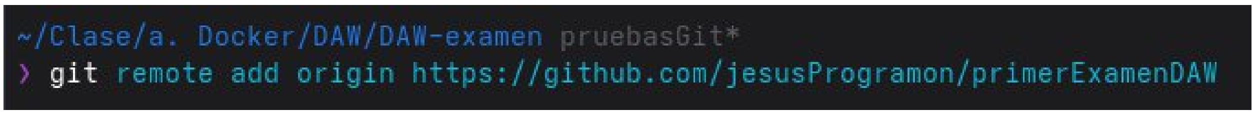
Add license — No license
Licenses explain how others can use your code. About licenses

Inicializamos git



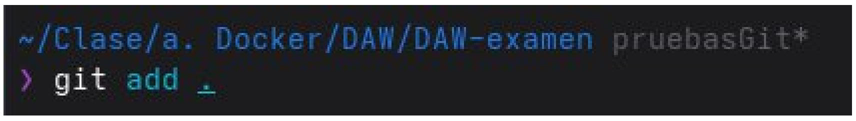
```
~/Clase/a. Docker/DAW/DAW-examen 2m 8s
❯ git init
Inicializado repositorio Git vacío en /home/thinkpad/Clase/a. Docker/DAW/DAW-examen/.git/
```

Añadimos el origen (repositorio)



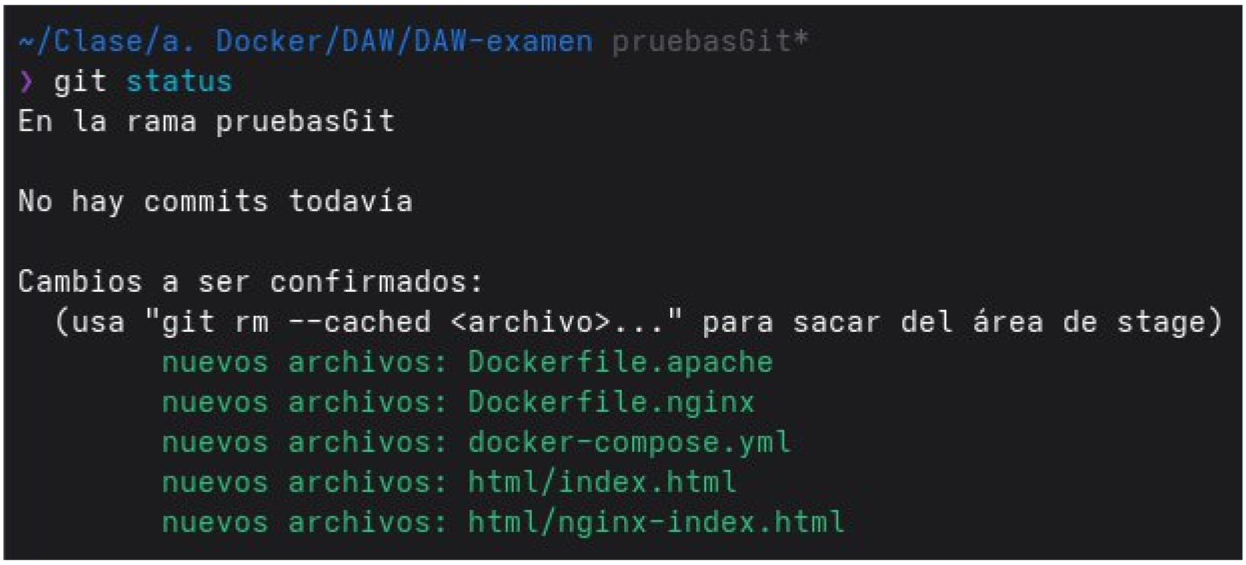
```
~/Clase/a. Docker/DAW/DAW-examen pruebasGit*
❯ git remote add origin https://github.com/jesusProgramon/primerExamenDAW
```

Añadimos los archivos



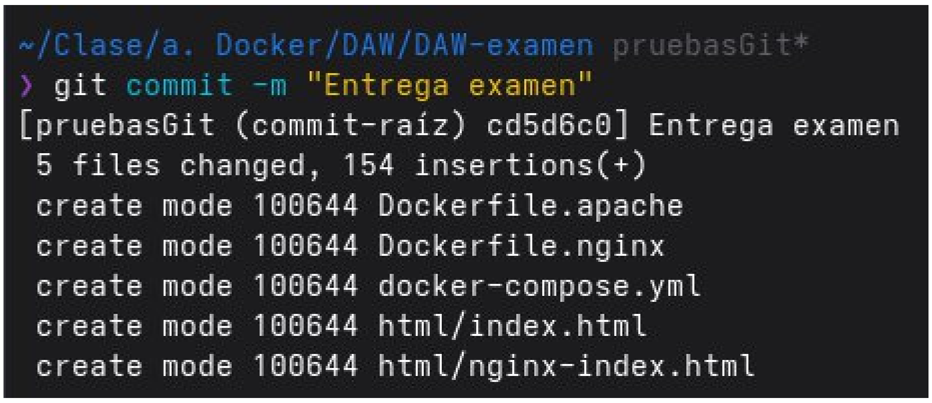
```
~/Clase/a. Docker/DAW/DAW-examen pruebasGit*
❯ git add .
```

Comprobamos que está todo bien



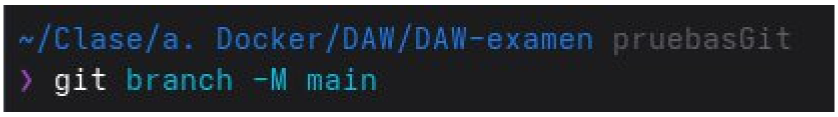
```
~/Clase/a. Docker/DAW/DAW-examen pruebasGit*
❯ git status
En la rama pruebasGit

No hay commits todavía

Cambios a ser confirmados:
  (usa "git rm --cached <archivo>..." para sacar del área de stage)
        nuevos archivos: Dockerfile.apache
        nuevos archivos: Dockerfile.nginx
        nuevos archivos: docker-compose.yml
        nuevos archivos: html/index.html
        nuevos archivos: html/nginx-index.html
```

Hacemos el commit



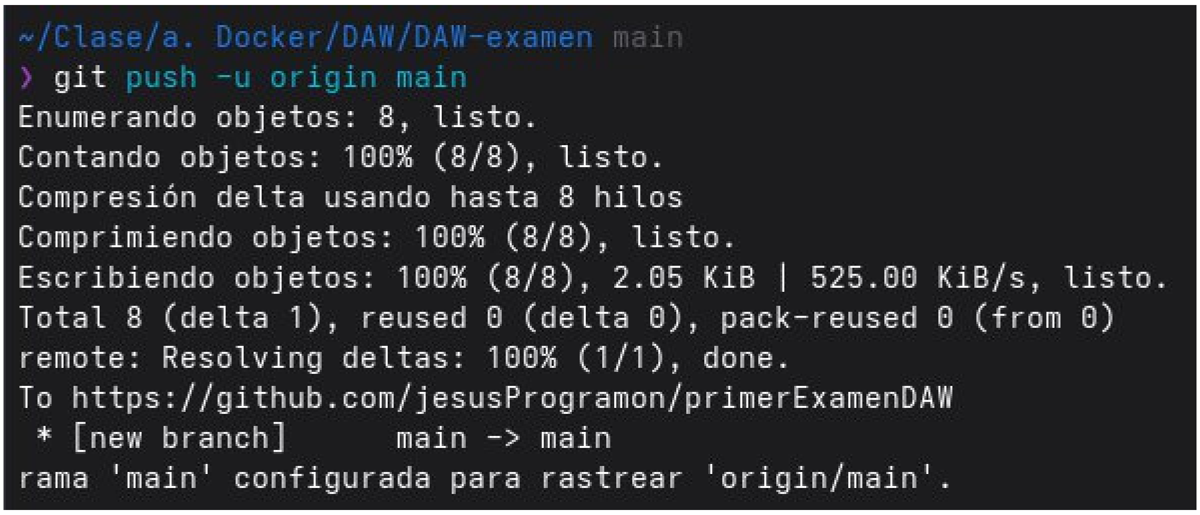
```
~/Clase/a. Docker/DAW/DAW-examen pruebasGit*
❯ git commit -m "Entrega examen"
[pruebasGit (commit-raíz) cd5d6c0] Entrega examen
 5 files changed, 154 insertions(+)
 create mode 100644 Dockerfile.apache
 create mode 100644 Dockerfile.nginx
 create mode 100644 docker-compose.yml
 create mode 100644 html/index.html
 create mode 100644 html/nginx-index.html
```

Nos situamos en la rama main



```
~/Clase/a. Docker/DAW/DAW-examen pruebasGit
❯ git branch -M main
```

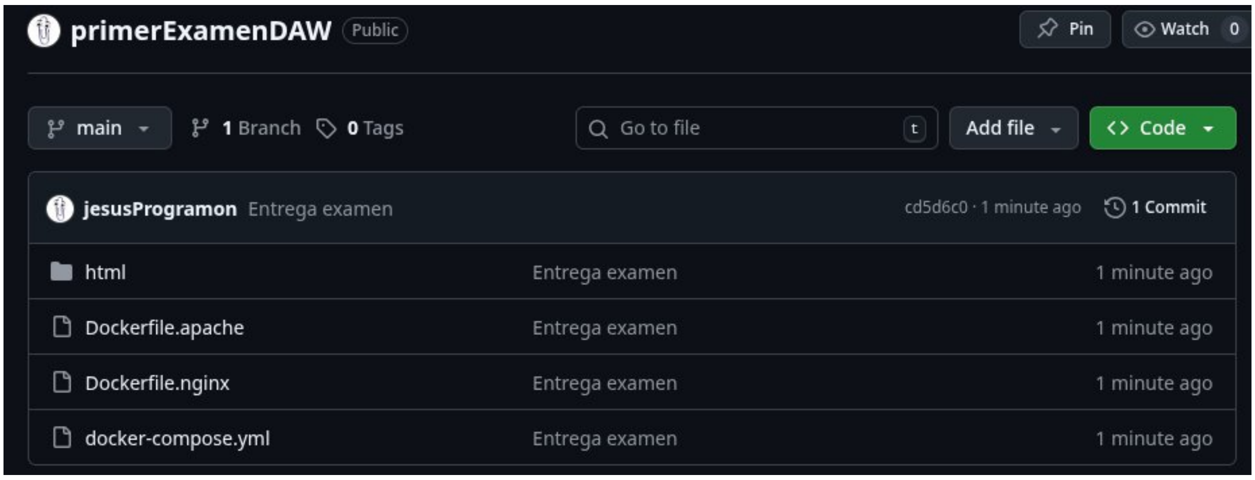
Hacemos el git push para subirlo



```
~/Clase/a. Docker/DAW/DAW-examen main
❯ git push -u origin main
Enumerando objetos: 8, listo.
Contando objetos: 100% (8/8), listo.
Compresión delta usando hasta 8 hilos
Comprimiendo objetos: 100% (8/8), listo.
Escribiendo objetos: 100% (8/8), 2.05 KiB | 525.00 KiB/s, listo.
Total 8 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/jesusProgramon/primerExamenDAW
 * [new branch]      main -> main
rama 'main' configurada para rastrear 'origin/main'.
```

Comprobamos que está subido en nuestro repositorio

https://github.com/jesusProgramon/primerExamenDAW



```
jesusProgramon Entrega examen          cd5d6c0 · 1 minute ago   1 Commit
html                   Entrega examen   1 minute ago
Dockerfile.apache      Entrega examen   1 minute ago
Dockerfile.nginx       Entrega examen   1 minute ago
docker-compose.yml     Entrega examen   1 minute ago
```