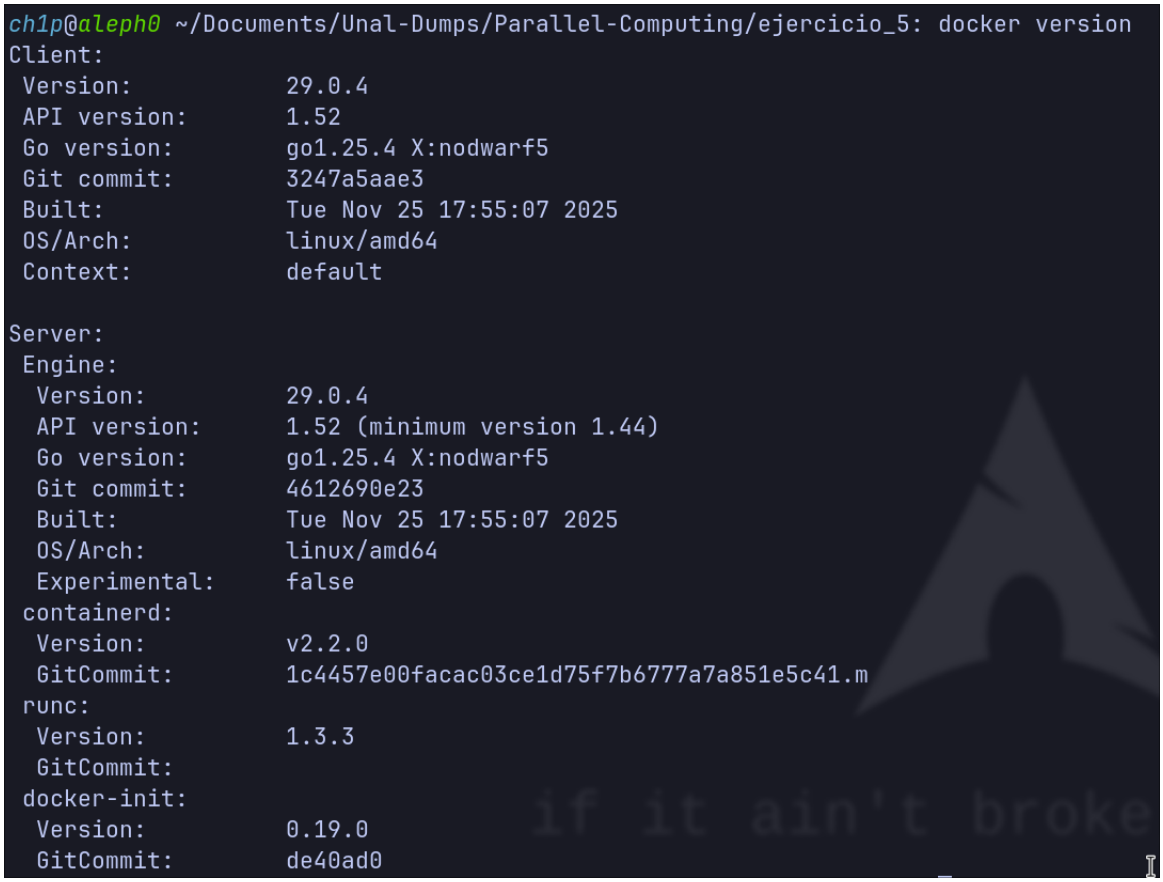
Ejercicio 5

Nombres REDACTED

A continuación, se presenta la evidencia de la correcta instalación de `Docker`, requisito para la realización del presente ejercicio.



Ejercicio 4

Para el desarrollo del ejercicio 4, se inició preparando los archivos necesarios. En primer lugar, se creó el directorio `some-content`, dentro del cual se generó el archivo `index.html` con el siguiente contenido:

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Using Nginx with Docker</title>
  </head>
  <body>
    <h1>Nginx server working</h1>
    <p>This content is just a dummy</p>
  </body>
</html>
```

Una vez preparados los archivos, se procedió a crear el primer contenedor mediante el siguiente comando:

```
docker run --name some-nginx \
  -v $(pwd)/some-content:/usr/share/nginx/html:ro \
  -d nginx
```

Este comando permite ejecutar un contenedor denominado `some-nginx` utilizando la imagen oficial de `nginx`. El parámetro `-v` monta el directorio `some-content` del host en la ruta `/usr/share/nginx/html` del contenedor, estableciendo el acceso en modo de solo lectura (*read-only*). Además, el parámetro `-d` indica que el contenedor se ejecute en segundo plano (*detached mode*).

La siguiente evidencia confirma que el contenedor se encuentra en ejecución:

```
ch1p@aleph0 ~/Documents/Unal-Dumps/Parallel-Computing/ejercicio_5/nginx-ejemplo/some-content: docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS     NAMES
b5c18db04666   nginx     "/docker-entrypoint.…"   About a minute ago   Up About a minute   80/tcp    some-nginx
```

Posteriormente, se procedió a crear una imagen personalizada de Nginx. Para ello, se creó el directorio `static-html-directory`, cuyo contenido es el mismo archivo HTML utilizado anteriormente. Luego, se generó el archivo `Dockerfile` con el siguiente contenido:

```
FROM nginx
COPY static-html-directory /usr/share/nginx/html
```

Este archivo especifica el uso de la imagen base oficial de `nginx` y copia el contenido HTML al directorio donde Nginx sirve los archivos estáticos.

A continuación, se construyó la imagen personalizada mediante el comando:

```
docker build -t some-content-nginx .
```

El uso de la opción `-t` permite asignar la etiqueta `some-content-nginx` a la imagen generada.

```
ch1p@aleph0 ~/Documents/Unal-Dumps/Parallel-Computing/ejercicio_5/nginx-ejemplo: docker build -t some-content-nginx .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  5.632kB
Step 1/2 : FROM nginx
 ---> 576306625d79
Step 2/2 : COPY static-html-directory /usr/share/nginx/html
 ---> 29f934087d36
Successfully built 29f934087d36
Successfully tagged some-content-nginx:latest
```

La correcta creación de la imagen se verificó utilizando el comando `docker images`:

```
ch1p@aleph0 ~/Documents/Unal-Dumps/Parallel-Computing/ejercicio_5/nginx-ejemplo: docker images
                                                                                    4 Info →  U In Use
IMAGE                      ID             DISK USAGE   CONTENT SIZE   EXTRA
nginx:latest               576306625d79       152MB            0B
some-content-nginx:latest  29f934087d36       152MB            0B
```

Posteriormente, se creó un contenedor a partir de la imagen personalizada:

```
docker run --name some-nginx -d -p 8081:80 some-content-nginx
```

En este caso, se mapea el puerto 8081 del host con el puerto 80 del contenedor, permitiendo el acceso al servidor web desde un navegador.

Ejercicio 6, 7 y 8

Posteriormente, se utilizó el archivo `compose.yaml` del repositorio `awesome-compose`. Para ello, se ejecutó el comando:

```
docker compose up -d
```

Este comando permite levantar los servicios definidos en el archivo `compose.yaml` en modo desacoplado. La creación de los contenedores fue validada mediante:

```
ch1p@aleph0 ~/Documents/Unal-Dumps/Parallel-Computing/ejercicio_5/nginx-golang: docker ps -a
CONTAINER ID   IMAGE                  COMMAND                  CREATED          STATUS          PORTS                                          NAMES
9858f6e619fc   nginx                  "/docker-entrypoint.…"   2 minutes ago    Up 2 minutes    0.0.0.0:80->80/tcp, [::]:80->80/tcp            nginx-go
lang-proxy-1
e41963cc5a7f   nginx-golang-backend   "/code/bin/backend"      2 minutes ago    Up 2 minutes                                                   nginx-go
lang-backend-1
a7c0a2565645   some-content-nginx     "/docker-entrypoint.…"   20 minutes ago   Up 20 minutes   0.0.0.0:8081->80/tcp, [::]:8081->80/tcp        some-ngi
nx
```

Asimismo, se verificó la creación de la imagen correspondiente al backend con Golang:

```
ch1p@aleph0 ~/Documents/Unal-Dumps/Parallel-Computing/ejercicio_5/nginx-golang: docker images
                                                                                                  i Info →  U In Use
IMAGE                          ID             DISK USAGE   CONTENT SIZE   EXTRA
nginx-golang-backend:latest    8c1e2ac5b6a0       337MB             0B    U
nginx:latest                   576306625d79       152MB             0B    U
some-content-nginx:latest      29f934087d36       152MB             0B    U
```

Finalmente, se comprobó la ejecución correcta del servicio backend:

```
ch1p@aleph0 ~/Documents/Unal-Dumps/Parallel-Computing/ejercicio_5/nginx-golang: curl localhost:80

         ##         .
   ## ## ##        ==
 ## ## ## ## ##    ===
/"""""""""""""""""\___/ ===
{                     /  ===-
\______ O           __/
 \    \         __/
  \____\_______/


Hello from Docker!
```

Para determinar la versión de Golang utilizada, se ejecutó el comando:

```
ch1p@aleph0 ~/Documents/Unal-Dumps/Parallel-Computing/ejercicio_5/nginx-golang: docker exec -it e41963cc5a7f sh
/code # go version
go version go1.18.10 linux/amd64
```

En esta etapa, se observó que el servicio utilizaba la versión `Go 1.18.10`. Adicionalmente, se verificó el sistema operativo mediante el comando `cat /etc/os-release`:

```
/code # cat /etc/os-release
NAME="Alpine Linux"
ID=alpine
VERSION_ID=3.17.1
PRETTY_NAME="Alpine Linux v3.17"
HOME_URL="https://alpinelinux.org/"
BUG_REPORT_URL="https://gitlab.alpinelinux.org/alpine/aports/-/issues"
```

Confirmando así el uso de `Alpine Linux 3.17.1`.

Ejercicio 9 y 10

Se procedió a detener todos los contenedores previamente ejecutados. En primer lugar, se utilizaron los siguientes comandos:

```
docker compose down
docker stop some-nginx
```

El comando `docker compose down` permite detener y eliminar los contenedores, redes y recursos asociados a los servicios creados mediante `Docker Compose`. Por su parte, el comando `docker stop some-nginx` se utilizó para detener el contenedor creado previamente mediante `docker run`.

Ejercicio 11 y 12

Posteriormente, se modificó el archivo `Dockerfile` para utilizar explícitamente la versión `nginx:1.27.5-alpine`:

```
FROM nginx:1.27.5-alpine
COPY static-html-directory /usr/share/nginx/html
```

La nueva imagen fue construida utilizando:

```
docker build -t some-content-nginx .
```

```
ch1p@aleph0 ~/Documents/Unal-Dumps/Parallel-Computing/ejercicio_5/nginx-ejemplo: docker images

IMAGE                          ID             DISK USAGE   CONTENT SIZE   EXTRA
nginx-golang-backend:latest    8c1e2ac5b6a0        337MB             0B
nginx:1.27.5-alpine            6769dc3a703c       48.2MB             0B
nginx:latest                   576306625d79        152MB             0B
some-content-nginx:latest      d086527cdd56       48.2MB             0B
```

En esta figura se puede apreciar la razón por la cual se eligió Alpine como sistema operativo para el contenedor: como se observa, las `docker images` basadas en Alpine son considerablemente más livianas que las demás.

```
/ # nginx -v
nginx version: nginx/1.27.5
/ #
```

Ejercicio 13

De igual manera, se actualizó el archivo `compose.yaml` para utilizar Golang 1.24 sobre Alpine Linux:

```
backend:
  image: golang:1.24-alpine
  build:
    context: backend
    target: builder
```

Y el Dockerfile correspondiente:

```
FROM --platform=$BUILDPLATFORM golang:1.24-alpine AS builder
WORKDIR /code
```

Finalmente, tras ejecutar nuevamente:

```
docker compose up -d
```

Se verificó que el servicio backend se ejecuta correctamente utilizando `Golang 1.24` sobre `Alpine Linux`, cumpliendo con los requisitos establecidos en el ejercicio.

```
ch1p@aleph0 ~/Documents/Unal-Dumps/Parallel-Computing/ejercicio_5/nginx-golang/backer
/code # go version
go version go1.24.11 linux/amd64
/code # cat /etc/os-release
NAME="Alpine Linux"
ID=alpine
VERSION_ID=3.23.0
PRETTY_NAME="Alpine Linux v3.23"
HOME_URL="https://alpinelinux.org/"
BUG_REPORT_URL="https://gitlab.alpinelinux.org/alpine/aports/-/issues"
/code #
```