2021

Practica 3 - Dockerfile

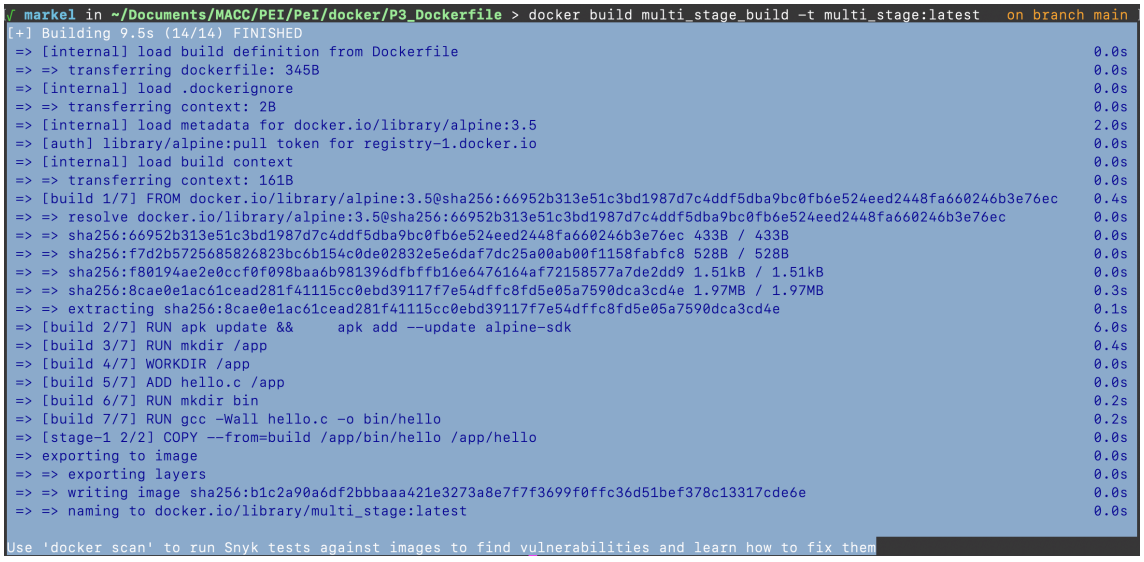
MARKEL ORALLO

Multi-Stage Builds

Creamos la imagen a partir del Dockerfile:

FROM alpine:3.5 AS build RUN apk update && \
apk add --update alpine-sdk RUN mkdir /app
WORKDIR /app
ADD hello.c /app
RUN mkdir bin
RUN gcc -Wall hello.c -o bin/hello
Lightweight image returned as final product
FROM alpine:3.5
COPY --from=build /app/bin/hello /app/hello
CMD /app/hello

Comprobar la imagen final mediante docker image ls (comprobar su tamaño) y describir el proceso. ¿Cuantos Stages tiene?



Primero crea una imagen que compila el programa hello.c y lo convierte a un binario. Para ello, se descarga “alpine-sdk” paquete que no necesitaremos para hacer funcionar el binario, tan solo para compilarlo. Después crea una imagen nueva en la que solo copia el binario compilado del stage anterior. Por lo tanto, tiene dos stages, uno para compilar el programa en c y otro para crear la imagen con el programa de c ya compilado.

Si lanzamos la imagen resultante y abrimos una terminal interactiva dentro de ella podemos comprobar que el contenedor no tiene instalado el compilador gcc que es parte del paquete “alpine-sdk” y se instala en el stage 1.

```
✓ markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker run -it multi_stage /bin/sh                    on branch main
/ # apk info
WARNING: Ignoring APKINDEX.c51f8f92.tar.gz: No such file or directory
WARNING: Ignoring APKINDEX.d09172fd.tar.gz: No such file or directory
musl
busybox
alpine-baselayout
alpine-keys
libressl2.4-libcrypto
libressl2.4-libssl
zlib
apk-tools
scanelf
musl-utils
libc-utils
/ #
```

De ahí deducimos que la imagen resultante no contiene los paquetes instalados en el stage 1 y por lo tanto es mas pequeña que una que la imagen que resultaría de hacer las mismas operaciones en un solo stage.

Por último, podemos comprobar que si lanzamos el contenedor el programa hello.c se ejecuta:

```
✓ markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker run multi_stage                                on branch main
Hello, World!%
```

```
✓ markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker images                                          on branch main
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
multi_stage   latest    b1c2a90a6df2   38 seconds ago   4.01MB
```

Escribir y generar el Dockerfile

Crea una carpeta “myImage” y crea un fichero con el nombre Dockerfile dentro de esta carpeta.
Contenido del Dockerfile:

FROM centos:7
RUN yum update –y
RUN yum install -y wget

Esto sirve como receta para una imagen basada en centos:7, que tiene todos sus paquetes por defecto actualizados y wget instalado. Genera la imagen con el nombre myimage . Compruebe que su nueva imagen existe (docker image ls)

```
✓ markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > mkdir myImage                                          on branch main
✓ markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > vim myImage/Dockerfile                                 on branch main
✓ markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker build myImage -t myimage                        on branch main
[+] Building 37.4s (8/8) FINISHED
 => [internal] load build definition from Dockerfile                                                          0.0s
 => => transferring dockerfile: 99B                                                                           0.0s
 => [internal] load .dockerignore                                                                             0.0s
 => => transferring context: 2B                                                                               0.0s
 => [internal] load metadata for docker.io/library/centos:7                                                   2.1s
 => [auth] library/centos:pull token for registry-1.docker.io                                                 0.0s
 => [1/3] FROM docker.io/library/centos:7@sha256:9d4bcbbb213dfd745b58be38b13b996ebb5ac315fe75711bd618426a630e0987  7.5s
 => => resolve docker.io/library/centos:7@sha256:9d4bcbbb213dfd745b58be38b13b996ebb5ac315fe75711bd618426a630e0987  0.0s
 => => sha256:9d4bcbbb213dfd745b58be38b13b996ebb5ac315fe75711bd618426a630e0987 1.20kB / 1.20kB                 0.0s
 => => sha256:dead07b4d8ed7e29e98de0f4504d87e8880d4347859d839686a31da35a3b532f 529B / 529B                     0.0s
 => => sha256:eeb6ee3f44bd0b5103bb561b4c16bcb82328cfe5809ab675bb17ab3a16c517c9 2.75kB / 2.75kB                 0.0s
 => => sha256:2d473b07cdd5f0912cd6f1a703352c82b512407db6b05b43f2553732b55df3bc 76.10MB / 76.10MB               3.9s
 => => extracting sha256:2d473b07cdd5f0912cd6f1a703352c82b512407db6b05b43f2553732b55df3bc                      3.3s
 => [2/3] RUN yum update -y                                                                                  24.7s
 => [3/3] RUN yum install -y wget                                                                             1.5s
 => exporting to image                                                                                        1.4s
 => => exporting layers                                                                                       1.4s
 => => writing image sha256:77bf60af076c8a0f6c909b10d482d8c5859aa22f3383ada275384271ca953af9                  0.0s
 => => naming to docker.io/library/myimage                                                                    0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
✓ markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker images                                          on branch main
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
myimage       latest    77bf60af076c   12 seconds ago   572MB
multi_stage   latest    b1c2a90a6df2   22 minutes ago   4.01MB
✓ markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile >                                                       on branch main
```

Ejecutar la imagen en un contenedor y haga wget en ese contenedor.

```
✓ markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker run -it myimage:latest /bin/bash                on branch main
[root@718b5e52e214 /]# wget google.es
--2021-09-23 18:22:26--  http://google.es/
Resolving google.es (google.es)... 172.217.17.3, 2a00:1450:4003:80c::2003
Connecting to google.es (google.es)|172.217.17.3|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: http://www.google.es/ [following]
--2021-09-23 18:22:26--  http://www.google.es/
Resolving www.google.es (www.google.es)... 172.217.17.3, 2a00:1450:4003:806::2003
Reusing existing connection to google.es:80.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'index.html'

    [ <=>                                                                   ] 15,812      --.-K/s   in 0.01s

2021-09-23 18:22:27 (1.40 MB/s) - 'index.html' saved [15812]

[root@718b5e52e214 /]#
```

Build Caché

Abra su Dockerfile y agregue otro paso RUN al final para instalar vim.
RUN yum install vim. Construya la imagen de nuevo como en el ejemplo anterior; ¿para qué pasos se utiliza la caché?

```
 3 FROM centos:7
 2 RUN yum update -y
 1 RUN yum install -y wget
 0 RUN yum install -y vim
```

```
✓ markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > vim myImage/Dockerfile                                on branch main
✓ markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker build myImage -t myimage                       on branch main
[+] Building 9.7s (9/9) FINISHED
 => [internal] load build definition from Dockerfile                                                                0.0s
 => => transferring dockerfile: 121B                                                                                0.0s
 => [internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 2B                                                                                     0.0s
 => [internal] load metadata for docker.io/library/centos:7                                                         1.3s
 => [auth] library/centos:pull token for registry-1.docker.io                                                       0.0s
 => [1/4] FROM docker.io/library/centos:7@sha256:9d4bcbbb213dfd745b58be38b13b996ebb5ac315fe75711bd618426a630e0987   0.0s
 => CACHED [2/4] RUN yum update -y                                                                                  0.0s
 => CACHED [3/4] RUN yum install -y wget                                                                            0.0s
 => [4/4] RUN yum install -y vim                                                                                    7.4s
 => exporting to image                                                                                              1.0s
 => => exporting layers                                                                                             1.0s
 => => writing image sha256:15fde99e52da8cfc51a2a86c0637941edb4016116baf0a88e2ff439beb48d321                        0.0s
 => => naming to docker.io/library/myimage                                                                          0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

Utiliza la cache para los pasos que hemos realizado en el primer build de la imagen. Para la orden *RUN yum install -y vim no utiliza la cache. Esto se debe a que esta operación no se a realizado en el build anterior.*

Construya la imagen de nuevo; ¿qué pasos utiliza esta vez la caché?

```
✓ markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker build myImage -t myimage                       on branch main
[+] Building 0.8s (8/8) FINISHED
 => [internal] load build definition from Dockerfile                                                                0.0s
 => => transferring dockerfile: 121B                                                                                0.0s
 => [internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 2B                                                                                     0.0s
 => [internal] load metadata for docker.io/library/centos:7                                                         0.7s
 => [1/4] FROM docker.io/library/centos:7@sha256:9d4bcbbb213dfd745b58be38b13b996ebb5ac315fe75711bd618426a630e0987   0.0s
 => CACHED [2/4] RUN yum update -y                                                                                  0.0s
 => CACHED [3/4] RUN yum install -y wget                                                                            0.0s
 => CACHED [4/4] RUN yum install -y vim                                                                             0.0s
 => exporting to image                                                                                              0.0s
 => => exporting layers                                                                                             0.0s
 => => writing image sha256:15fde99e52da8cfc51a2a86c0637941edb4016116baf0a88e2ff439beb48d321                        0.0s
 => => naming to docker.io/library/myimage                                                                          0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

Si volvememos a construir la imagen vemos que tiene cacheados todos los pasos ya que los a realizado en el build anterior.

Cambia el orden de los dos comandos RUN para instalar wget y vim en el Dockerfile, y construye una vez más la imagen. ¿Qué pasos se almacenan en la memoria caché esta vez?

```
3 FROM centos:7
2 RUN yum update -y
1 RUN yum install -y vim
0 RUN yum install -y wget
```

```
markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker build myImage -t myimage                    on branch main
[+] Building 11.0s (9/9) FINISHED
 => [internal] load build definition from Dockerfile                                                          0.0s
 => => transferring dockerfile: 121B                                                                          0.0s
 => [internal] load .dockerignore                                                                             0.0s
 => => transferring context: 2B                                                                               0.0s
 => [internal] load metadata for docker.io/library/centos:7                                                   1.3s
 => [auth] library/centos:pull token for registry-1.docker.io                                                 0.0s
 => [1/4] FROM docker.io/library/centos:7@sha256:9d4bcbbb213dfd745b58be38b13b996ebb5ac315fe75711bd618426a630e0987  0.0s
 => CACHED [2/4] RUN yum update -y                                                                            0.0s
 => [3/4] RUN yum install -y vim                                                                              7.3s
 => [4/4] RUN yum install -y wget                                                                             1.3s
 => exporting to image                                                                                        1.0s
 => => exporting layers                                                                                       1.0s
 => => writing image sha256:681858493c7404731a543dcdd88e2144f2cef8e25163b3692a6604703bfaa29d                  0.0s
 => => naming to docker.io/library/myimage                                                                    0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

En esta caso vemos que al cambiar el orden de los comandos la cache deja de funcionar en la instalación de vim. Esto se debe a que la cache esperaba la instalación de wget y al ver que la siguiente orden se instala vim, se desactiva para el resto del build.

History Command

Ejecuta docker image history a la imagen creada en el paso anterior y observa la información devuelta

```
markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker image history myimage:latest                on branch main
IMAGE          CREATED          CREATED BY                                      SIZE      COMMENT
681858493c74   8 minutes ago    RUN /bin/sh -c yum install -y wget # buildkit   120MB     buildkit.dockerfile.v0
<missing>      8 minutes ago    RUN /bin/sh -c yum install -y vim # buildkit    175MB     buildkit.dockerfile.v0
<missing>      17 minutes ago   RUN /bin/sh -c yum update -y # buildkit         248MB     buildkit.dockerfile.v0
<missing>      8 days ago       /bin/sh -c #(nop)  CMD ["/bin/bash"]            0B
<missing>      8 days ago       /bin/sh -c #(nop)  LABEL org.label-schema.sc…   0B
<missing>      8 days ago       /bin/sh -c #(nop) ADD file:b3ebbe8bd304723d4…   204MB
```

Cambia las instrucciones de instalación de wget y vim para que se instalen con un único comando: RUN yum install -y wget vim

Genera la imagen de nuevo y llama al comando history. ¿Qué ha cambiado?

```
markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > vim myImage/Dockerfile                             on branch main
markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker build myImage -t myimage                    on branch main
[+] Building 23.8s (7/7) FINISHED
 => [internal] load build definition from Dockerfile                                                          0.0s
 => => transferring dockerfile: 102B                                                                          0.0s
 => [internal] load .dockerignore                                                                             0.0s
 => => transferring context: 2B                                                                               0.0s
 => [internal] load metadata for docker.io/library/centos:7                                                   0.7s
 => [1/3] FROM docker.io/library/centos:7@sha256:9d4bcbbb213dfd745b58be38b13b996ebb5ac315fe75711bd618426a630e0987  0.0s
 => CACHED [2/3] RUN yum update -y                                                                            0.0s
 => [3/3] RUN yum install -y vim wget                                                                        21.9s
 => exporting to image                                                                                        1.1s
 => => exporting layers                                                                                       1.1s
 => => writing image sha256:b729308206ce7a08a3049d67d3afc9ad1c61f3dd73858aebb714392e4b5791ec                  0.0s
 => => naming to docker.io/library/myimage                                                                    0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker image history myimage:latest                on branch main
IMAGE          CREATED          CREATED BY                                      SIZE      COMMENT
b729308206ce   7 seconds ago    RUN /bin/sh -c yum install -y vim wget # bui…   176MB     buildkit.dockerfile.v0
<missing>      19 minutes ago   RUN /bin/sh -c yum update -y # buildkit         248MB     buildkit.dockerfile.v0
<missing>      8 days ago       /bin/sh -c #(nop)  CMD ["/bin/bash"]            0B
<missing>      8 days ago       /bin/sh -c #(nop)  LABEL org.label-schema.sc…   0B
<missing>      8 days ago       /bin/sh -c #(nop) ADD file:b3ebbe8bd304723d4…   204MB
```

Podemos ver que al ejecutar la instalación de wget y vim en la misma orden RUN la imagen tiene una capa menos. Antes la cada una de las ordenes RUN generaba una capa diferente. Ahora, al incluir la instalación de vim y wget en la misma orden RUN se forma una única capa.

Configurar comandos por defecto

Añade la siguiente línea al Dockerfile de la práctica anterior: CMD ["ping", "127.0.0.1", "-c", "5"]. Genera de nuevo la imagen y pon en marcha el contenedor. ¿Que obtienes?

```
[✓ markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > vim myImage/Dockerfile                                    on branch main
[✓ markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker build myImage -t myimage                           on branch main
[+] Building 1.5s (8/8) FINISHED
 => [internal] load build definition from Dockerfile                                                                    0.0s
 => => transferring dockerfile: 139B                                                                                    0.0s
 => [internal] load .dockerignore                                                                                       0.0s
 => => transferring context: 2B                                                                                         0.0s
 => [internal] load metadata for docker.io/library/centos:7                                                             1.4s
 => [auth] library/centos:pull token for registry-1.docker.io                                                           0.0s
 => [1/3] FROM docker.io/library/centos:7@sha256:9d4bcbbb213dfd745b58be38b13b996ebb5ac315fe75711bd618426a630e0987       0.0s
 => CACHED [2/3] RUN yum update -y                                                                                      0.0s
 => CACHED [3/3] RUN yum install -y vim wget                                                                            0.0s
 => exporting to image                                                                                                  0.0s
 => => exporting layers                                                                                                 0.0s
 => => writing image sha256:8e4621d6af916106ed7222e5cfeb7077efa9a09255dbb887d1f7f1e1b2656e3c                            0.0s
 => => naming to docker.io/library/myimage                                                                              0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
[✓ markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker run myimage                                        on branch main
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.020 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.029 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.055 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.056 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.056 ms

--- 127.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4082ms
rtt min/avg/max/mdev = 0.020/0.043/0.056/0.016 ms
```

Al añadir la orden CMD el comando que se ejecuta al iniciar el contenedor es el siguiente: *ping 127.0.0.1 -c 5*
Por lo tanto, al lanzar el contenedor se ejecutará el comando anterior y al finalizar el contenedor se caerá.

Ahora ejecuta el siguiente comando al poner en marcha el contenedor: $docker container run myimage echo "hello world“ .¿Qué ha pasado esta vez?

```
[✓ markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker run myimage echo "Hello World"                    on branch main
Hello World
```

Al pasarle el comando *echo “Hello World”* al final del comando docker run se sobrescribe la orden CMD del fichero de Dockerfile. De esta manera, el comando que se ejecutará en al iniciarse el contenedor será: *echo “Hello World”*

Cambia la última línea del Dockerfile por: ENTRYPOINT ["ping"].Genera y pon en marcha la nueva imagen en un nuevo contenedor. ¿Qué pasa?

```
[✓ markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > vim myImage/Dockerfile                                    on branch main
[✓ markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker build myImage -t myimage                           on branch main
[+] Building 1.4s (8/8) FINISHED
 => [internal] load build definition from Dockerfile                                                                      0.0s
 => => transferring dockerfile: 122B                                                                                      0.0s
 => [internal] load .dockerignore                                                                                         0.0s
 => => transferring context: 2B                                                                                           0.0s
 => [internal] load metadata for docker.io/library/centos:7                                                               1.3s
 => [auth] library/centos:pull token for registry-1.docker.io                                                             0.0s
 => [1/3] FROM docker.io/library/centos:7@sha256:9d4bcbbb213dfd745b58be38b13b996ebb5ac315fe75711bd618426a630e0987         0.0s
 => CACHED [2/3] RUN yum update -y                                                                                        0.0s
 => CACHED [3/3] RUN yum install -y vim wget                                                                              0.0s
 => exporting to image                                                                                                    0.0s
 => => exporting layers                                                                                                   0.0s
 => => writing image sha256:8d406eaa6fb9d8bfda65fa48fa8272dd53f7e79c5953a9fb5af6dea4131e4b58                              0.0s
 => => naming to docker.io/library/myimage                                                                                0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
[✓ markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker run myimage                                         on branch main
Usage: ping [-aAbBdDfhLnOqrRUvV64] [-c count] [-i interval] [-I interface]
            [-m mark] [-M pmtudisc_option] [-l preload] [-p pattern] [-Q tos]
            [-s packetsize] [-S sndbuf] [-t ttl] [-T timestamp_option]
            [-w deadline] [-W timeout] [hop1 ...] destination
Usage: ping -6 [-aAbBdDfhLnOqrRUvV] [-c count] [-i interval] [-I interface]
             [-l preload] [-m mark] [-M pmtudisc_option]
             [-N nodeinfo_option] [-p pattern] [-Q tclass] [-s packetsize]
             [-S sndbuf] [-t ttl] [-T timestamp_option] [-w deadline]
             [-W timeout] destination
```

Al iniciar el contenedor se ejecuta el comando *ping*. Podemos ver el output del comando en pantalla.

Prueba a poner en marcha mendiante este comando:
$docker container run myimage 127.0.0.1

```
[x markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker run myimage 127.0.0.1                              on branch main
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.018 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.055 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.054 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.055 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.051 ms
^C
--- 127.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4124ms
rtt min/avg/max/mdev = 0.018/0.046/0.055/0.016 ms
```

Ahora al tener el entrypoint definido como ping en el Dockerfile lo que pongamos al final del comando docker run pasara como parámetro del comando ping.

¿Puedes describir las diferencias entre ENTRYPOINT y CMD?

ENTRYPOINT defina la aplicación que se pondrá en marcha al iniciar el contendor y CMD define el comando que se ejecutara al inciar el contenedor. En caso de tener ambas ordenes en el fichero Dockerfile CMD pasará a ser parámetro del comando definido en ENTRYPOINT.

Cambia la última línea del Dockerfile por:
ENTRYPOINT ["ping", "-c", "3"] CMD ["127.0.0.1"]
Genera la imagen y pon en marcha el contenedor (sin pasarle comandos) ¿Te ha dado error? ¿Cómo interpreta el valor de CMD?

```
markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > vim myImage/Dockerfile                    on branch main
markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker build myImage -t myimage           on branch main
[+] Building 1.4s (8/8) FINISHED
 => [internal] load build definition from Dockerfile                                          0.0s
 => => transferring dockerfile: 151B                                                          0.0s
 => [internal] load .dockerignore                                                             0.0s
 => => transferring context: 2B                                                               0.0s
 => [internal] load metadata for docker.io/library/centos:7                                   1.3s
 => [auth] library/centos:pull token for registry-1.docker.io                                 0.0s
 => [1/3] FROM docker.io/library/centos:7@sha256:9d4bcbbb213dfd745b58be38b13b996ebb5ac315fe75711bd618426a630e0987  0.0s
 => CACHED [2/3] RUN yum update -y                                                            0.0s
 => CACHED [3/3] RUN yum install -y vim wget                                                  0.0s
 => exporting to image                                                                        0.0s
 => => exporting layers                                                                       0.0s
 => => writing image sha256:284a62bfc0ef9877ed9d782f72ca6470fa3d411637955bb704784e47a1dcdfe9  0.0s
 => => naming to docker.io/library/myimage                                                    0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker run myimage                      on branch main
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.034 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.058 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.054 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2054ms
rtt min/avg/max/mdev = 0.034/0.048/0.058/0.013 ms
markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile >                                        on branch main
```

No da error. Interpreta el CMD como parámetro del ENTRYPOINT.

Pon en marcha el contenedor:
$docker container run myimage 8.8.8.8 ¿Qué ha pasado esta vez?

```
markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile > docker run myimage 8.8.8.8              on branch main
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=37 time=14.9 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=37 time=21.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=37 time=13.3 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
rtt min/avg/max/mdev = 13.330/16.635/21.583/3.565 ms
markel in ~/Documents/MACC/PEI/PeI/docker/P3_Dockerfile >                                        on branch main
```

Al pasarle 8.8.8.8 en el commando docker run, docker a sobrescrito la orden CMD que habíamos definido en el Dockerfile y a sustituido 127.0.0.1 por 8.8.8.8 .