

# Docker

**1. Crea un Dockerfile que partiendo de una imagen PHP genera una imagen que:**

**Copia una aplicación en PHP a un directorio del contenedor. Esta aplicación se debe copiar directamente desde un directorio del anfitrión. Para facilitar las cosas, debe de ser una aplicación sencilla que no emplee bases de datos (ya que si no también habría que instalar un MySQL).**

Creamos un directorio docker/practica1

Clonamos el proyecto github de <https://github.com/fullstack-superdev/simple-php-website>

git clone <https://github.com/fullstack-superdev/simple-php-website>

```
(kali㉿kali)-[~/docker/practica1]
└─$ git clone https://github.com/fullstack-superdev/simple-php-website.git
Cloning into 'simple-php-website' ...
remote: Enumerating objects: 117, done.
remote: Counting objects: 100% (117/117), done.
remote: Compressing objects: 100% (60/60), done.
remote: Total 117 (delta 51), reused 117 (delta 51), pack-reused 0
Receiving objects: 100% (117/117), 18.22 KiB | 2.60 MiB/s, done.
Resolving deltas: 100% (51/51), done.
```

Creamos el fichero Dockerfile en el mismo directorio con el siguiente contenido:

# Usa una imagen base de PHP con Apache

FROM php:7.4-apache

WORKDIR /home/kali/docker/practica1/

# Copia el contenido del directorio del anfitrión al directorio /var/www/html del contenedor

COPY simple-php-website/ /var/www/html/

# Exponer el puerto 80 para que la aplicación PHP pueda ser accesible desde fuera del contenedor

EXPOSE 80

# CMD predeterminado para el contenedor

CMD ["apache2-foreground"]

Creamos la imagen del contenedor con el comando:

```
docker build -t mi-aplicacion-php .
```

```
(kali㉿kali)-[~/docker/practica1]
└─$ sudo docker build -t mi-aplicacion-php .
[sudo] password for kali:
Sending build context to Docker daemon 127.5kB
Step 1/4 : FROM php:7.4-apache
7.4-apache: Pulling from library/php
a603fa5e3b41: Pull complete
c428f1a49423: Pull complete
156740b07ef8: Extracting [=====>] 41.22MB/91.63MB
fb5a4c8af82f: Download complete
25f85b498fd5: Download complete
9b233e420ac7: Download complete
fe42347c4ecf: Download complete
d14eb2ed1e17: Download complete
66d98f73acb6: Download complete
d2c43c5efbc8: Download complete
ab590b48ea47: Download complete
80692ae2d067: Download complete
05e465aaa99a: Download complete
```

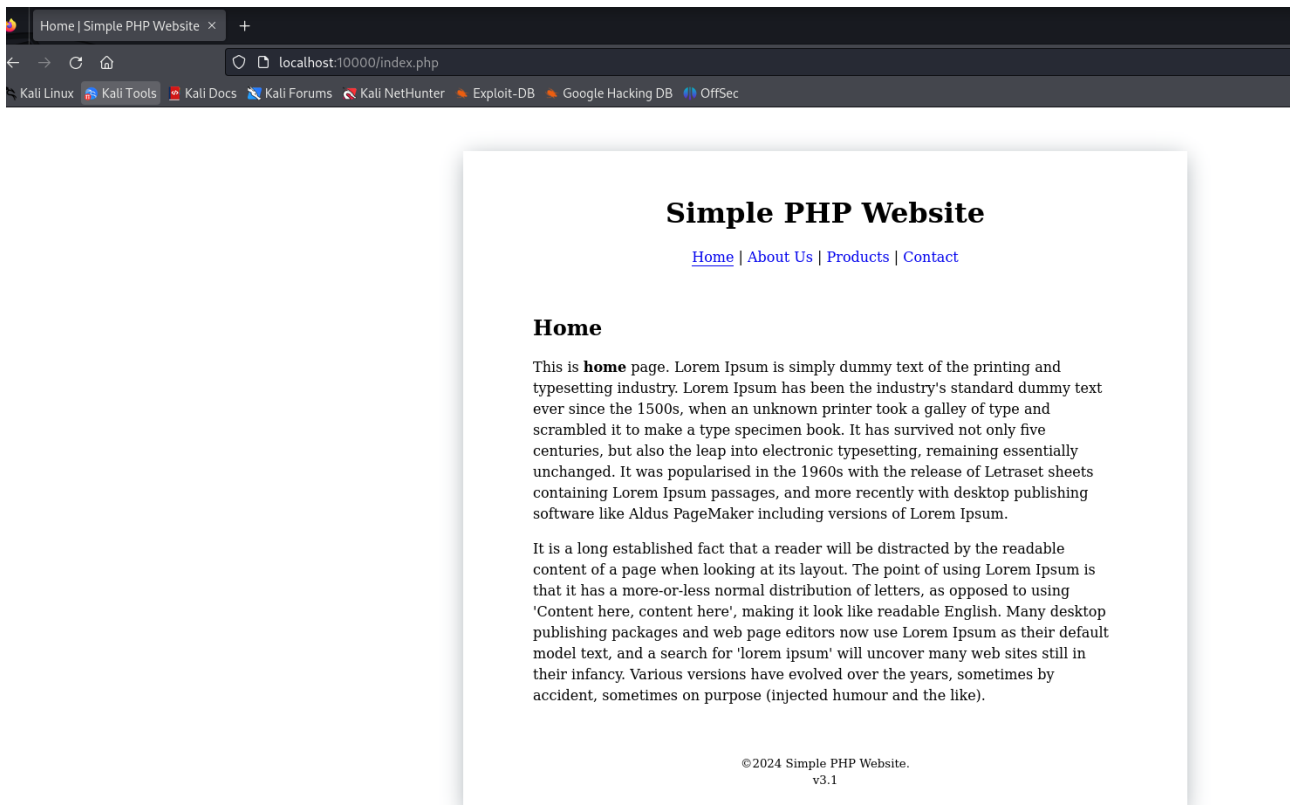
```
(kali㉿kali)-[~/docker/practica1]
└─$ sudo docker build -t mi-aplicacion-php .
Sending build context to Docker daemon 127.5kB
Step 1/5 : FROM php:7.4-apache
    -> 20a3732f422b
Step 2/5 : WORKDIR /home/kali/docker/practica1/
    -> Running in fb38fc7b564b
Removing intermediate container fb38fc7b564b
    -> 6454a55a709e
Step 3/5 : COPY url-shortener/ /var/www/html/
    -> 33e7cd7d2c4f
Step 4/5 : EXPOSE 80
    -> Running in 96676ecf3c8f
Removing intermediate container 96676ecf3c8f
    -> 42e71288997e
Step 5/5 : CMD ["apache2-foreground"]
    -> Running in 684fa8003b39
Removing intermediate container 684fa8003b39
    -> 1e776bd7355f
Successfully built 1e776bd7355f
Successfully tagged mi-aplicacion-php:latest
```

Ejecuta el contenedor, mapeando el puerto 10000 del host al puerto 80 del contenedor. Al contenedor le llamamos mi-contenedor-php

```
docker run -d -p 10000:80 --name mi-contenedor-php mi-aplicacion-php
```

```
(kali㉿kali)-[~]
└─$ docker run -d -p 10000:80 --name mi-contenedor-php mi-aplicacion-php
95e6e8d9f55826fa372e92eb0af68268a066dfacd3cc7f512866bb85428581d4
```

Probamos a conectarnos al puerto 10000 de nuestro anfitrión para ver si nuestra aplicación funciona:



Tuve un problema al conectar a la web con el fichero .htaccess que tiene en el raiz el servidor apache, me tuve que conectar al docker usando este comando y moviendo el .htaccess a un directorio old

`sudo docker exec -it 9ed7f1172a3b /bin/bash`

```
(kali㉿kali)-[~/docker/practical1]
$ sudo docker exec -it 9ed7f1172a3b /bin/bash
root@9ed7f1172a3b:/home/kali/docker/practical1# cd /var/www/html/
root@9ed7f1172a3b:/var/www/html# ls
content includes index.php readme.md template
root@9ed7f1172a3b:/var/www/html# mkdir old
root@9ed7f1172a3b:/var/www/html# mv .htaccess old/
root@9ed7f1172a3b:/var/www/html#
```

## **2. Crea un Dockerfile que partiendo de una imagen Ubuntu genera una imagen que:**

**A. Instala Apache, de forma que se exponga el puerto 80.**

**B. Instala PHP.**

**C. Copia una aplicación web en PHP al directorio de Apache que expone las páginas web.**

**Esta aplicación se debe descargar automáticamente mediante algún comando como git clone o curl. Para facilitar las cosas, debe de ser una aplicación sencilla que no emplee bases de datos (ya que si no también habría que instalar un MySQL).**

Creemos un directorio practica2 donde creamos un fichero Dockerfile con este contenido:

```
# Usa una imagen base de Ubuntu
```

```
FROM ubuntu:latest
```

```
# Instala Apache y PHP
```

```
RUN apt-get update && \
```

```
    apt-get install -y apache2 php libapache2-mod-php git && \
```

```
    rm -rf /var/lib/apt/lists/*
```

```
# Exponer el puerto 80 para Apache
```

```
EXPOSE 80
```

```
# Copia la aplicación web PHP al directorio de Apache
```

```
RUN rm -rf /var/www/html/*
```

```
RUN git clone https://github.com/fullstack-superdev/simple-php-website/var/www/html/
```

```
# CMD predeterminado para el contenedor
```

```
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

Creamos la segunda imagen con el comando:

```
sudo docker build -t mi-aplicacion2-php .
```

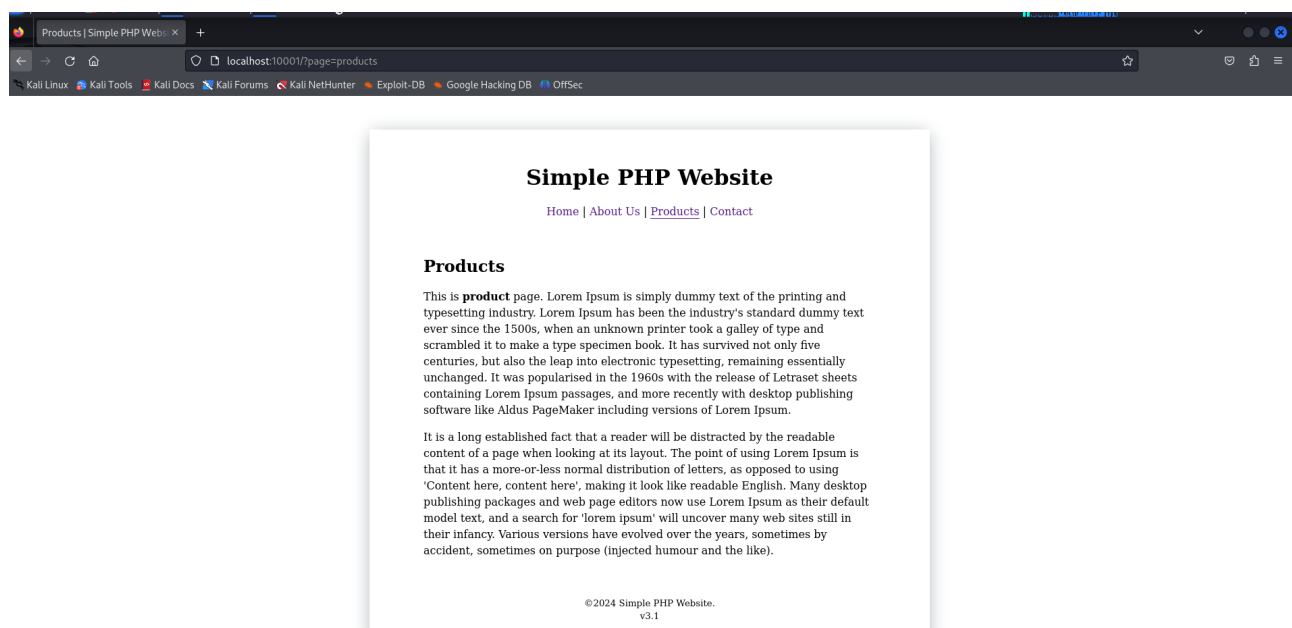
```
(kali@kali)-[~/docker/practica2]
└─$ sudo docker build -t mi-aplicacion2-php .
Sending build context to Docker daemon  2.56kB
Step 1/7 : FROM ubuntu:latest
--> bf3dc08bfd0
Step 2/7 : RUN echo "nameserver 1.1.1.1" > /etc/resolv.conf
--> Running in 1c5b262b9bfc
Removing intermediate container 1c5b262b9bfc
--> 7518d1636eac
Step 3/7 : RUN apt-get update && apt-get install -y apache2 php libapache2-mod-php git && rm -rf /var/lib/apt/lists/*
--> Running in a32c3b81248a
Get:1 http://security.ubuntu.com/ubuntu noble-security InRelease [89.7 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:3 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [26.0 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [10.3 kB]
Get:5 http://archive.ubuntu.com/ubuntu noble-updates InRelease [89.7 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble-backports InRelease [89.7 kB]
Get:7 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages [19.3 MB]
Get:8 http://archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [331 kB]
Get:9 http://archive.ubuntu.com/ubuntu noble/restricted amd64 Packages [117 kB]
Get:10 http://archive.ubuntu.com/ubuntu noble/main amd64 Packages [1808 kB]
Get:11 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [19.4 kB]
Get:12 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [31.5 kB]
Get:13 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [4224 B]
```

Iniciamos el docker a partir de esta imagen donde mapeamos el puerto 10001 del anfitrión al 80 del docker con el comando.

```
sudo docker run -d -p 10001:80 --name mi-contenedor2-php mi-aplicacion2-php
```

```
(kali@kali)-[~/docker/practica2]
└─$ sudo docker run -d -p 10001:80 --name mi-contenedor2-php mi-aplicacion2-php
ae8853e8753341f8c5ed585dfceec5435083f2be7322fe2f6e5aaf6c9d0582e3
(kali@kali)-[~/docker/practica2]
└─$
```

Ahora probamos a conectarnos al puerto 10001 desde el anfitrión.



### 3. Crea un contenedor para cada una de esas imágenes y verifica que funciona. Para y borra dicho contenedor.

Una vez probamos los contenedores los paramos con el comando:

```
sudo docker stop 9ed7f1172a3b
```

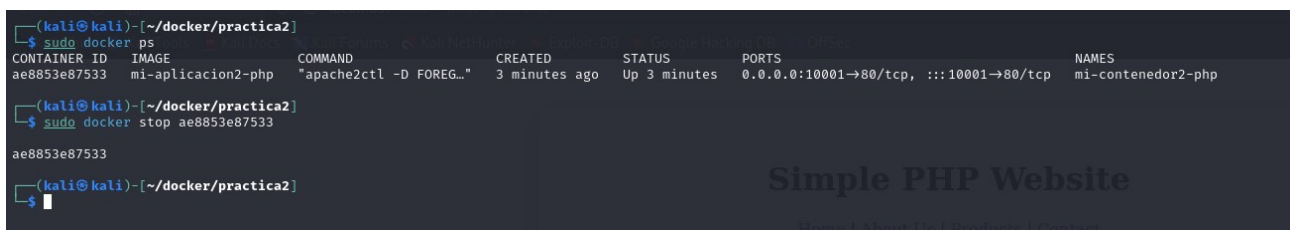
```
sudo docker stop ae8853e87533
```

Por ejemplo en el último caso es:

```
(kali㉿kali)-[~/docker/practica2]
└─$ sudo docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
ae8853e87533   mi-aplicacion2-php  "apachectl -D FOREG..."  3 minutes ago  Up 3 minutes  0.0.0.0:10001→80/tcp, :::10001→80/tcp  mi-contenedor2-php

(kali㉿kali)-[~/docker/practica2]
└─$ sudo docker stop ae8853e87533
ae8853e87533

(kali㉿kali)-[~/docker/practica2]
└─$
```



### 4. Emplea un comando para lanzar 20 contenedores de la segunda imagen, cada uno mapeado en un puerto distinto del anfitrión. Cuando veas que funcionan, para y borra dichos contenedores.

Dentro del directorio practica2 creamos un bashfile con el siguiente código:

```
—(kali㉿kali)-[~/docker/practica2]
```

```
└─$ more createDockers.sh
```

```
#!/bin/bash
```

```
port=10000
```

```
for i in 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20; do
```

```
    port=$((port+1))
```

```
    echo $port
```

```
    echo $i
```

```
    docker run -d -p $port:80 mi-aplicacion2-php
```

```
done
```

le damos permisos de ejecución y lo lanzamos con sudo

```
chmod +x createDockers.sh
```

Ejecutamos el script

```
(kali㉿kali)-[~/docker/practica2]
$ sudo sh createDockers.sh
[sudo] password for kali:
10001
1
2b1939e9020b7fc3cd17e81a51f6f4b00b0084778a9dcecb98b34f199cd057c8
10002
2
0b5d65632edfd7000311a69015137fb00ca472e80296939c021007bbfaa66f38
10003
3
b416bc2331a64259a862dbdd87d438998ae06168e5059bd7e3f72c2aca9e3a96
10004
4
4eaa24b7cfcae877b26439f6d3d661af7eb2d5910fbb72191473f0ccd5aace80d
10005
5
04f9e76aa40015427aa4b5537d36303c9008ac69159ec5abb176c1dda1f0d352
10006
6
f8c7ea0c7906149831eea66793dc5848d3f67c31c98891b5fba1322820150c27
10007
7
bc5ec5b96dc81656dded2b09a88cc885740eefcc0f45171395a52669c7f9e9ff
10008
8
1774404a3b62dfa405bb5b538c18ff8c4e21df83ab08d631e05072eb8d187f78
10009
9
5db00c801827fc71c2fb29538adee2479061c1cd4207f960155520b4c9a4db96
10010
10
eabcb6b13eaac80ff866163ebd40ee1603ae5df8812dfc00f868a6cd593212e7
10011
11
136724ea32a98ba35c00d7f0465dd4aa03ae60f2312567f147c116e7fd059376
10012
12
24ea92539e2b3e06f265a6e33f24d56c4a3d737d80cb441baf8cfafc959d2b4e
10013
13
554757c9d42a9ed42a6d24748cd29fa20165220c01adae893a23dc5a46c89139
10014
14
4beda3b0f81827be39c29c02e4758d6a8e653f65d432f5bba18c617cc1de590f
10015
15
40912a22b3b9bc5b7c64fa977524d365ea70ed58828e85c4bcbd879de94db4dc
10016
16
```

Comprobamos que están corriendo los contenedores:

```

(kali@kali)-[~/docker/practica2]
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
eee884532de9   mi-aplicacion2-php  "apache2ctl -D FOREG..." 2 minutes ago  Up 2 minutes  0.0.0.0:10020→80/tcp, :::10020→80/tcp  cool_driscoll
73aae299cff0   mi-aplicacion2-php  "apache2ctl -D FOREG..." 2 minutes ago  Up 2 minutes  0.0.0.0:10019→80/tcp, :::10019→80/tcp  ecstatic_goldwasser
69c81af312a2   mi-aplicacion2-php  "apache2ctl -D FOREG..." 2 minutes ago  Up 2 minutes  0.0.0.0:10018→80/tcp, :::10018→80/tcp  interesting_jones
94c0771e4e9d   mi-aplicacion2-php  "apache2ctl -D FOREG..." 2 minutes ago  Up 2 minutes  0.0.0.0:10017→80/tcp, :::10017→80/tcp  bold_lehmann
1abd3a235f8a   mi-aplicacion2-php  "apache2ctl -D FOREG..." 2 minutes ago  Up 2 minutes  0.0.0.0:10016→80/tcp, :::10016→80/tcp  sleepy_elgamel
40912a22b3b9   mi-aplicacion2-php  "apache2ctl -D FOREG..." 2 minutes ago  Up 2 minutes  0.0.0.0:10015→80/tcp, :::10015→80/tcp  funny_satoshi
4beda3b0f818   mi-aplicacion2-php  "apache2ctl -D FOREG..." 2 minutes ago  Up 2 minutes  0.0.0.0:10014→80/tcp, :::10014→80/tcp  hungry_mclean
554757c9d42a   mi-aplicacion2-php  "apache2ctl -D FOREG..." 2 minutes ago  Up 2 minutes  0.0.0.0:10013→80/tcp, :::10013→80/tcp  practical_goldstine
24ea92539e2b   mi-aplicacion2-php  "apache2ctl -D FOREG..." 2 minutes ago  Up 2 minutes  0.0.0.0:10012→80/tcp, :::10012→80/tcp  pensive_euclid
136724ea32a9   mi-aplicacion2-php  "apache2ctl -D FOREG..." 2 minutes ago  Up 2 minutes  0.0.0.0:10011→80/tcp, :::10011→80/tcp  fervent_cannon
eabcb6b13eaa   mi-aplicacion2-php  "apache2ctl -D FOREG..." 2 minutes ago  Up 2 minutes  0.0.0.0:10010→80/tcp, :::10010→80/tcp  elated_dhawan
5db00c801827   mi-aplicacion2-php  "apache2ctl -D FOREG..." 2 minutes ago  Up 2 minutes  0.0.0.0:10009→80/tcp, :::10009→80/tcp  hardcore_kepler
1774404a3b62   mi-aplicacion2-php  "apache2ctl -D FOREG..." 2 minutes ago  Up 2 minutes  0.0.0.0:10008→80/tcp, :::10008→80/tcp  charming_rosalind
bc5ec5b96dc8   mi-aplicacion2-php  "apache2ctl -D FOREG..." 2 minutes ago  Up 2 minutes  0.0.0.0:10007→80/tcp, :::10007→80/tcp  laughing_mendel
f8c7ea0c7906   mi-aplicacion2-php  "apache2ctl -D FOREG..." 2 minutes ago  Up 2 minutes  0.0.0.0:10006→80/tcp, :::10006→80/tcp  reverent_saha
04f9e76aa400   mi-aplicacion2-php  "apache2ctl -D FOREG..." 2 minutes ago  Up 2 minutes  0.0.0.0:10005→80/tcp, :::10005→80/tcp  romantic_faraday
4eaa24b7cfaf   mi-aplicacion2-php  "apache2ctl -D FOREG..." 2 minutes ago  Up 2 minutes  0.0.0.0:10004→80/tcp, :::10004→80/tcp  optimistic_austin
b416bc2331a6   mi-aplicacion2-php  "apache2ctl -D FOREG..." 2 minutes ago  Up 2 minutes  0.0.0.0:10003→80/tcp, :::10003→80/tcp  festive_pasteur
0b5d65632edf   mi-aplicacion2-php  "apache2ctl -D FOREG..." 2 minutes ago  Up 2 minutes  0.0.0.0:10002→80/tcp, :::10002→80/tcp  priceless_cartwright
2b1939e9020b   mi-aplicacion2-php  "apache2ctl -D FOREG..." 2 minutes ago  Up 2 minutes  0.0.0.0:10001→80/tcp, :::10001→80/tcp  exciting_elion

(kali@kali)-[~/docker/practica2]
$

```

Paramos todos los contenedores levantados:

`sudo docker stop $(docker ps -q)`

```

(kali@kali)-[~/docker/practica2]
$ sudo docker stop $(docker ps -q)
eee884532de9
73aae299cff0
69c81af312a2
94c0771e4e9d
1abd3a235f8a
40912a22b3b9
4beda3b0f818
554757c9d42a
24ea92539e2b
136724ea32a9
eabcb6b13eaa
5db00c801827
1774404a3b62
bc5ec5b96dc8
f8c7ea0c7906
04f9e76aa400
4eaa24b7cfaf
b416bc2331a6
0b5d65632edf
2b1939e9020b

(kali@kali)-[~/docker/practica2]
$

```



# **Git**

**Mantén un repositorio de Git público con:**

- **Memoria de la práctica.**
- **El fichero Dockerfile.**

**Haz un commit y un push del proyecto cada vez que superes uno de los ítems**