

MÓDULO: CONFIGURACIÓN DE ENTORNOS Y LABORATORIOS

Actividad de aprendizaje:

El siguiente taller tiene como objetivo que el Aprendiz conozca la tecnología de virtualización Docker y cómo a través de esta es posible desplegar aplicaciones y controlar sus características desde código.

<https://docs.docker.com/get-started/>

Para esta actividad el aprendiz deberá:

Realizar la creación de una app básica con Docker.

Al final deben presentar un informe donde se detalle la solución del taller con capturas de pantalla y conclusiones del ejercicio.

Comentarios: Se debe tener en cuenta la coherencia, cohesión y ortografía de los informes.

Creación de una app básica con Docker.

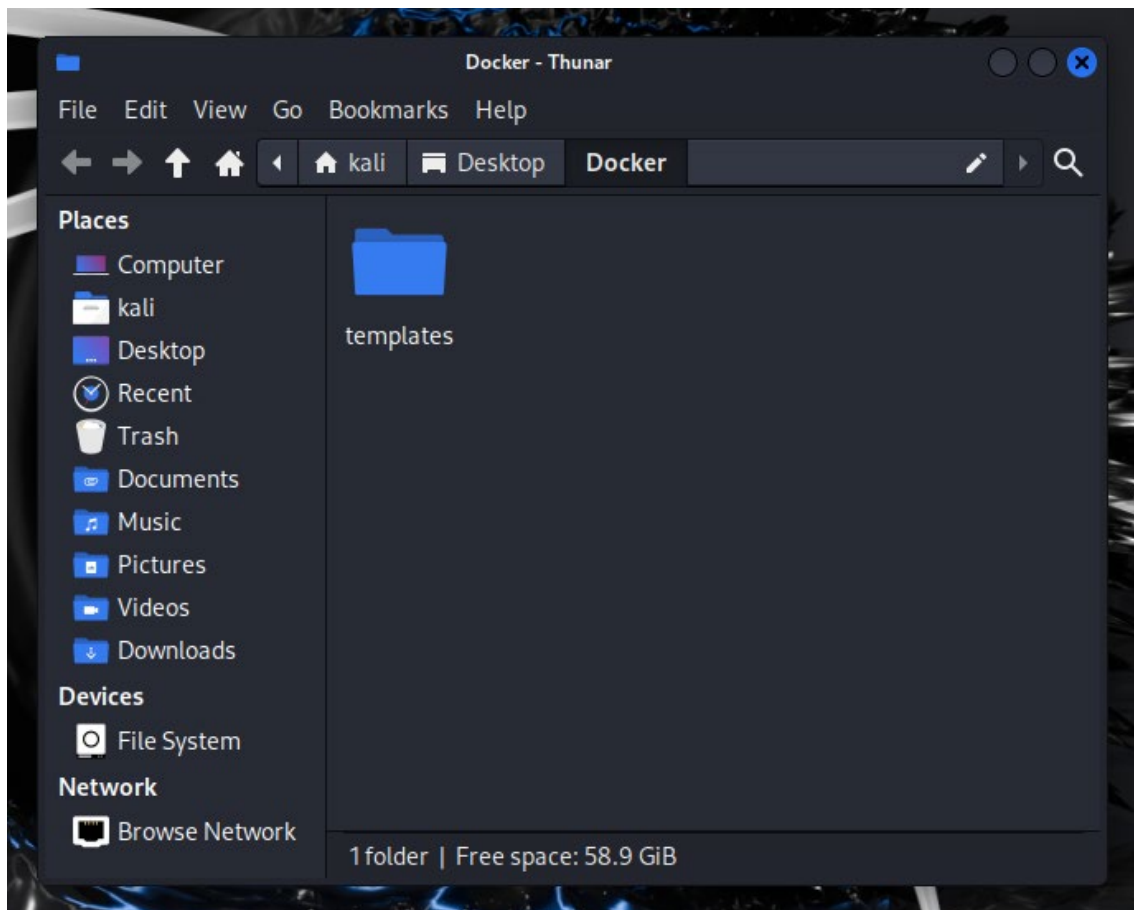
En primer lugar, tenemos en mente la creación de una app que nos permita ver imágenes aleatorias utilizando Flask.

Seguidamente creamos una carpeta con el nombre de la getting-started-app .

```
(kali㉿kali)-[~/Desktop]
$ mkdir Docker/

(kali㉿kali)-[~/Desktop]
$ mkdir Docker/templates

(kali㉿kali)-[~/Desktop]
$
```



Después de crear la carpeta Docker, añadiremos los siguientes ficheros dentro de esta: App.py, requirements.txt, Dockerfile y dentro de la subcarpeta templates, index.html.

```
(kali@kali) - [~/Desktop/Docker]
$ touch App.py

(kali@kali) - [~/Desktop/Docker]
$ touch requirements.txt

(kali@kali) - [~/Desktop/Docker]
$ touch Dockerfile

(kali@kali) - [~/Desktop/Docker]
$ cd templates

(kali@kali) - [~/Desktop/Docker/templates]
$ touch index.html

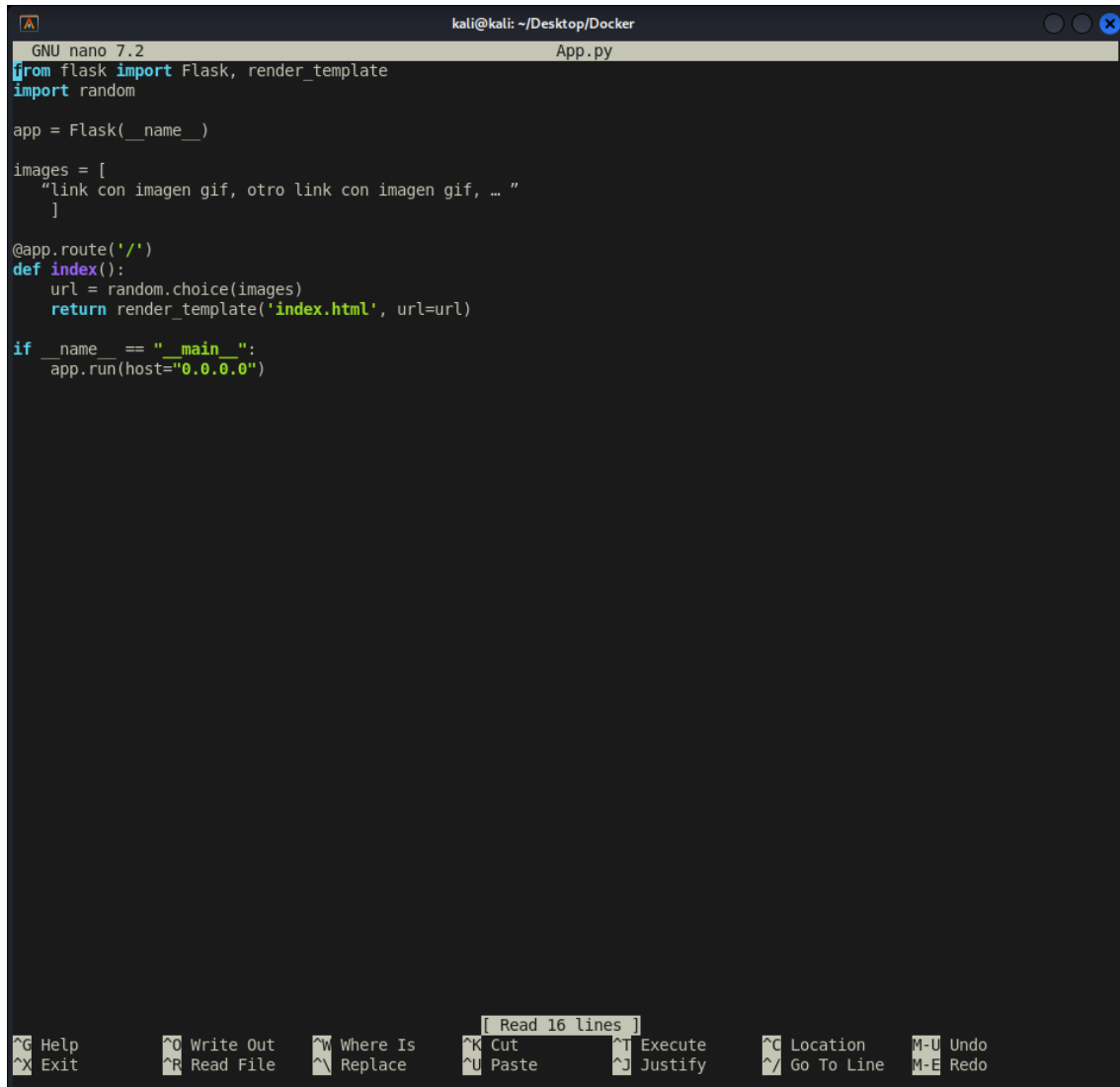
(kali@kali) - [~/Desktop/Docker/templates]
$ cd ..

(kali@kali) - [~/Desktop/Docker]
$ ls
App.py  Dockerfile  requirements.txt  templates

(kali@kali) - [~/Desktop/Docker]
$ ll
total 4
-rw-r--r-- 1 kali kali    0 Feb  1 06:59 App.py
-rw-r--r-- 1 kali kali    0 Feb  1 07:00 Dockerfile
-rw-r--r-- 1 kali kali    0 Feb  1 07:00 requirements.txt
drwxr-xr-x 2 kali kali 4096 Feb  1 07:00 templates

(kali@kali) - [~/Desktop/Docker]
$
```

El siguiente paso es ir uno a uno metiendo el código de cada fichero. Empezamos por el archivo App.py:



```
GNU nano 7.2 App.py
from flask import Flask, render_template
import random

app = Flask(__name__)

images = [
    "link con imagen gif, otro link con imagen gif, ..."
]

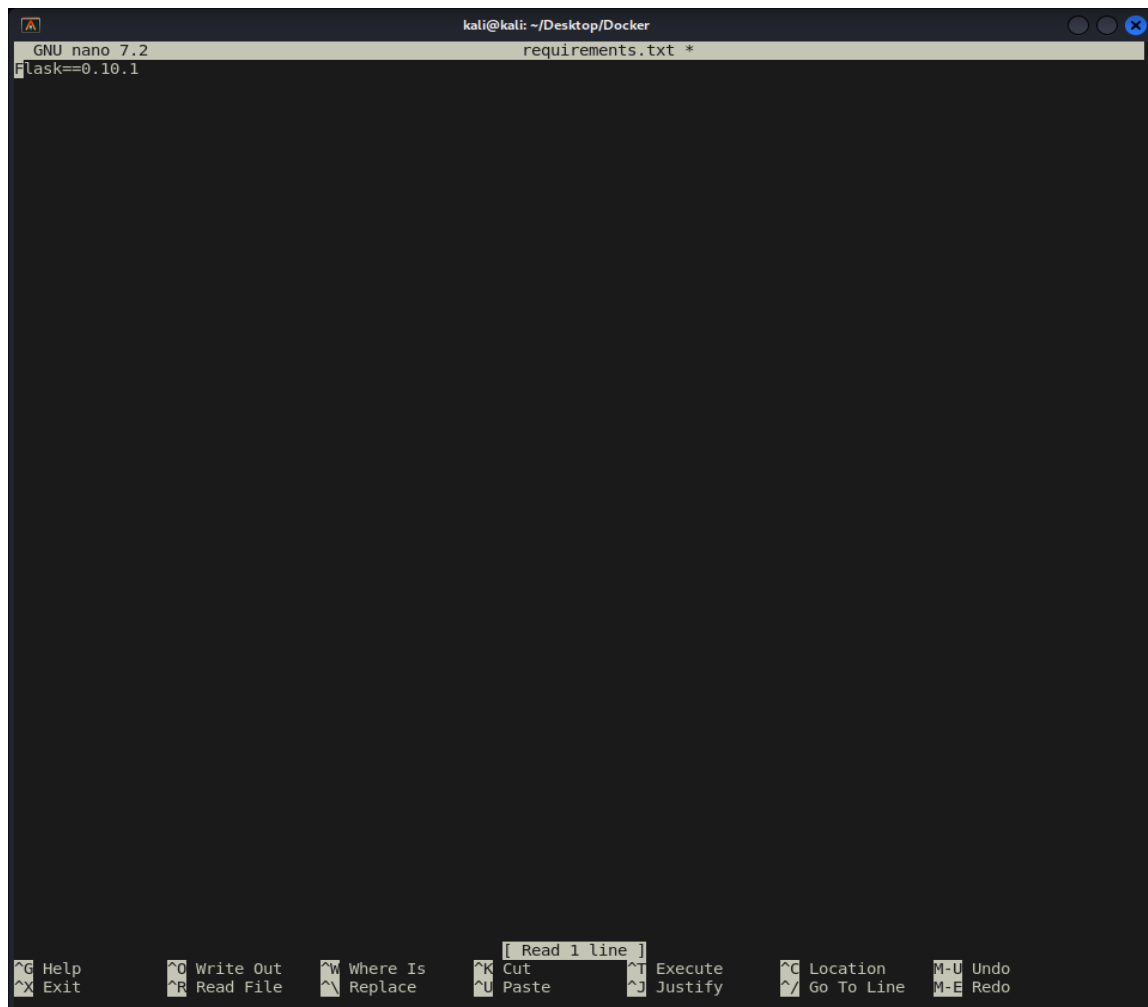
@app.route('/')
def index():
    url = random.choice(images)
    return render_template('index.html', url=url)

if __name__ == "__main__":
    app.run(host="0.0.0.0")
```

[Read 16 lines]

^G Help	^O Write Out	^W Where Is	^K Cut	^T Execute	^C Location	M-U Undo
^X Exit	^R Read File	^N Replace	^U Paste	^J Justify	^_ Go To Line	M-E Redo

Ahora el archivo requirements.txt:



The image shows a terminal window with the title bar "kali@kali: ~/Desktop/Docker". The window contains the GNU nano 7.2 text editor editing a file named "requirements.txt". The first line of the file is "flask==0.10.1". The bottom of the window displays the nano editor's help menu with the following options: ^G Help, ^O Write Out, ^W Where Is, [Read 1 line], ^C Location, M-U Undo, ^X Exit, ^R Read File, ^_ Replace, ^K Cut, ^T Execute, ^J Justify, ^_ Go To Line, and M-E Redo.

```
kali@kali: ~/Desktop/Docker
GNU nano 7.2 requirements.txt *
flask==0.10.1

[ Read 1 line ]
^G Help      ^O Write Out  ^W Where Is  [ Read 1 line ] ^C Location  M-U Undo
^X Exit      ^R Read File  ^_ Replace   ^K Cut        ^T Execute   ^J Justify
              ^_ Go To Line M-E Redo
```

Seguimos con el fichero `index.html`, en esta ocasión tenemos que movernos al directorio `Docker/templates`:

```
kali@kali: ~/Desktop/Docker/templates
GNU nano 7.2 index.html
<html>
<head>
  <style type="text/css">
    body {
      background: black;
      color: white;
    }
    div.container {
      max-width: 500px;
      margin: 100px auto;
      border: 20px solid white;
      padding: 10px;
      text-align: center;
    }
    h4 {
      text-transform: uppercase;
    }
  </style>
</head>
<body>
  <div class="container">
    <h4>Gif del dia</h4>
    
    <p><small>Cortesía: <a href="http://link_cualquiera">Estudiantes TdeA</a></small></p>
  </div>
</body>
</html>
```

[Read 27 lines]

Help	Write Out	Where Is	Cut	Execute	Location	M-U Undo
Exit	Read File	Replace	Paste	Justify	Go To Line	M-E Redo

El último archivo que queda por editar es Dockerfile.

Volvemos al directorio principal y ejecutamos nano Dockerfile.

```
kali@kali: ~/Desktop/Docker
GNU nano 7.2 Dockerfile
# Nuestra imagen base
FROM alpine:3.5

# instala Python pip
RUN apk add --update py2-pip

# instala los modulos de Python necesarios para la app
COPY requirements.txt /usr/src/app/
RUN pip install --no-cache-dir -r /usr/src/Docker/requirements.txt

# copia los archivos necesarios para la ejecución de la app
COPY app.py /usr/src/app/
COPY templates/index.html /usr/src/Docker/templates/

# especificamos el puerto que el contenedor expondrá
EXPOSE 5000

# ejecuta la app
CMD ["python", "/usr/src/Docker/App.py"]

[ Read 20 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^G Location   M-U Undo
^X Exit      ^R Read File  ^V Replace    ^U Paste      ^J Justify    ^_/ Go To Line  M-E Redo
```

Ahora tendremos que compilar la aplicación.

Con el comando: `sudo docker build -t getting-started-app .`

```
kali@kali: ~/Desktop/getting-started-app
(kali@kali)-[~/Desktop/getting-started-app]
$ sudo docker build -t getting-started-app .
Sending build context to Docker daemon 6.531MB
Step 1/6 : FROM node:18-alpine
--> c8eb770fbfac
Step 2/6 : WORKDIR /app
--> Using cache
--> 9742c63782db
Step 3/6 : COPY . .
--> Using cache
--> c2beaf0b7ae2
Step 4/6 : RUN yarn install --production
--> Running in 3d9d97b4f0f9
yarn install v1.22.19
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
[4/4] Building fresh packages...
Done in 13.75s.
Removing intermediate container 3d9d97b4f0f9
--> 72a7ecc11420
Step 5/6 : CMD ["node", "src/index.js"]
--> Running in 5bf4a864d43d
Removing intermediate container 5bf4a864d43d
--> d0a503af2c18
```

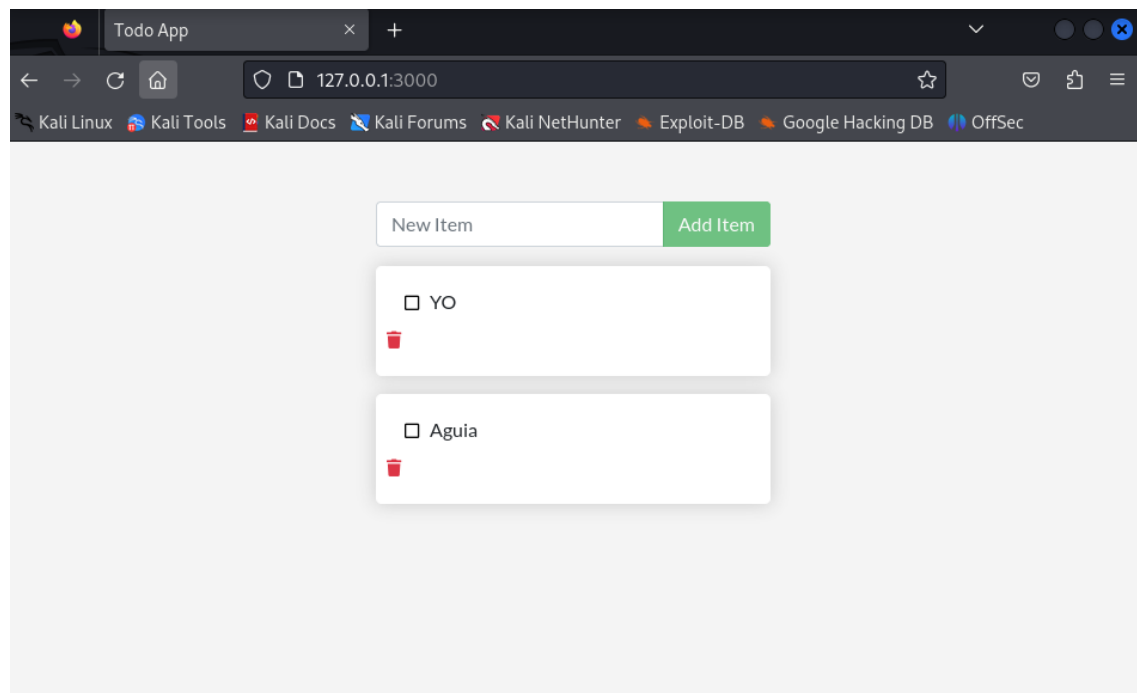
Después de compilar la app tenemos que ejecutarla con el siguiente comando:

```
Sudo docker run -dp 127.0.0.1:3000:3000 getting-started-app
```

```
(kali㉿kali)-[~/Desktop/getting-started-app]
$ sudo docker run -dp 127.0.0.1:3000:3000 getting-started-app
d241a5c01e51c5d9bfc35d83379cd7006b058f572965cf7cae806bf56e48ab4d
```

En teoría, el Docker tiene que estar en funcionamiento para ver si es así. Entonces abriremos nuestro navegador web en localhost y el puerto que hemos puesto por defecto.

<http://127.0.0.1:3000/>



El Docker funciona correctamente.