

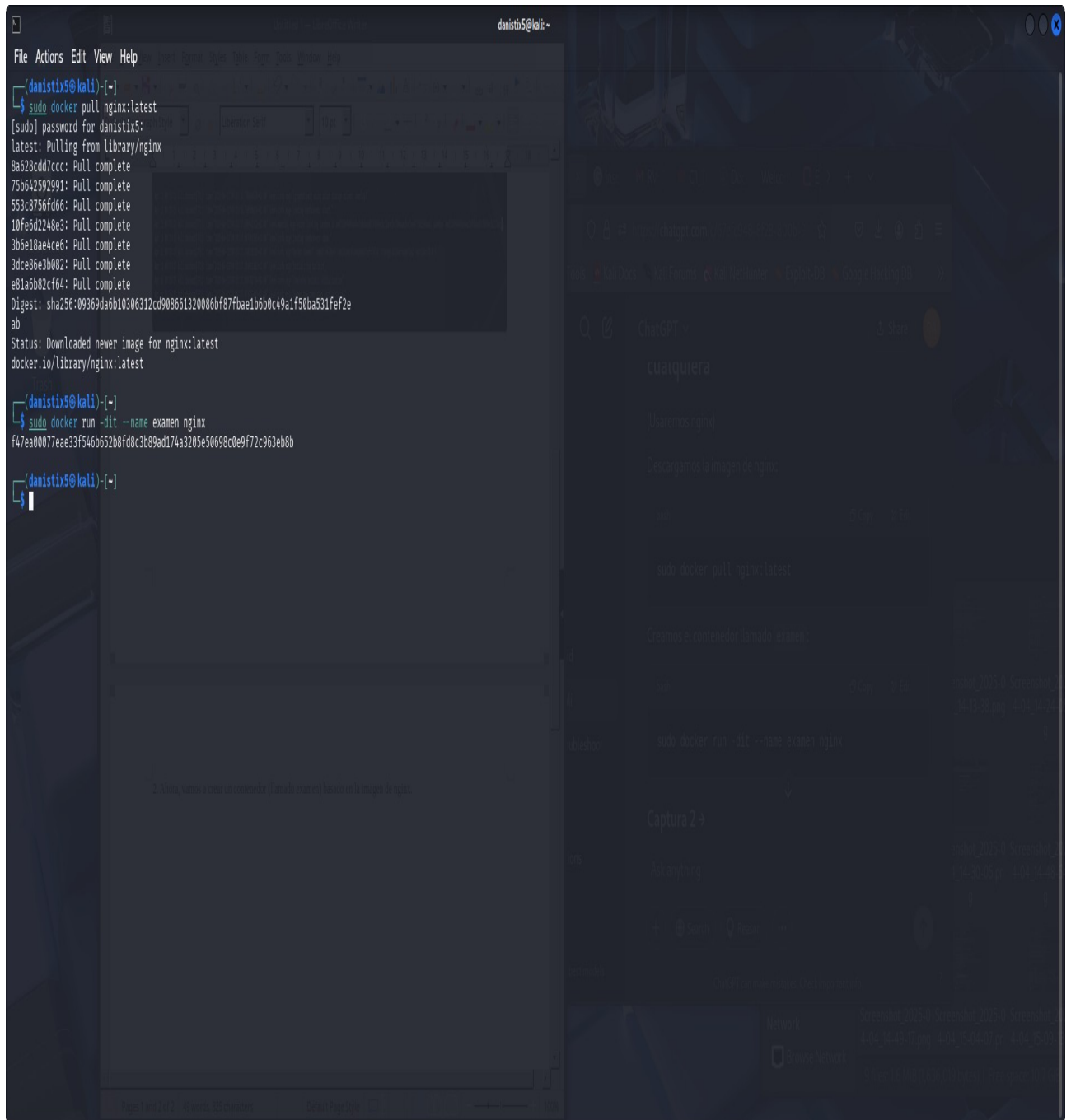
```
danistix@kali:~$ docker --version
Docker version 28.0.4, build b8034c0

(danistix@kali)~$ sudo systemctl status docker
[sudo] password for danistix:
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset:
   Active: active (running) since Fri 2025-04-11 09:35:17 CEST; 32min ago
 Invocation: 7ee476b6145f462f9fa0b23851a595e2
 TriggeredBy: ● docker.socket
   Docs: https://docs.docker.com
 Main PID: 771 (dockerd)
   Tasks: 9
  Memory: 99M (peak: 101M)
    CPU: 964ms
   CGroup: /system.slice/docker.service
           └─771 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/cb

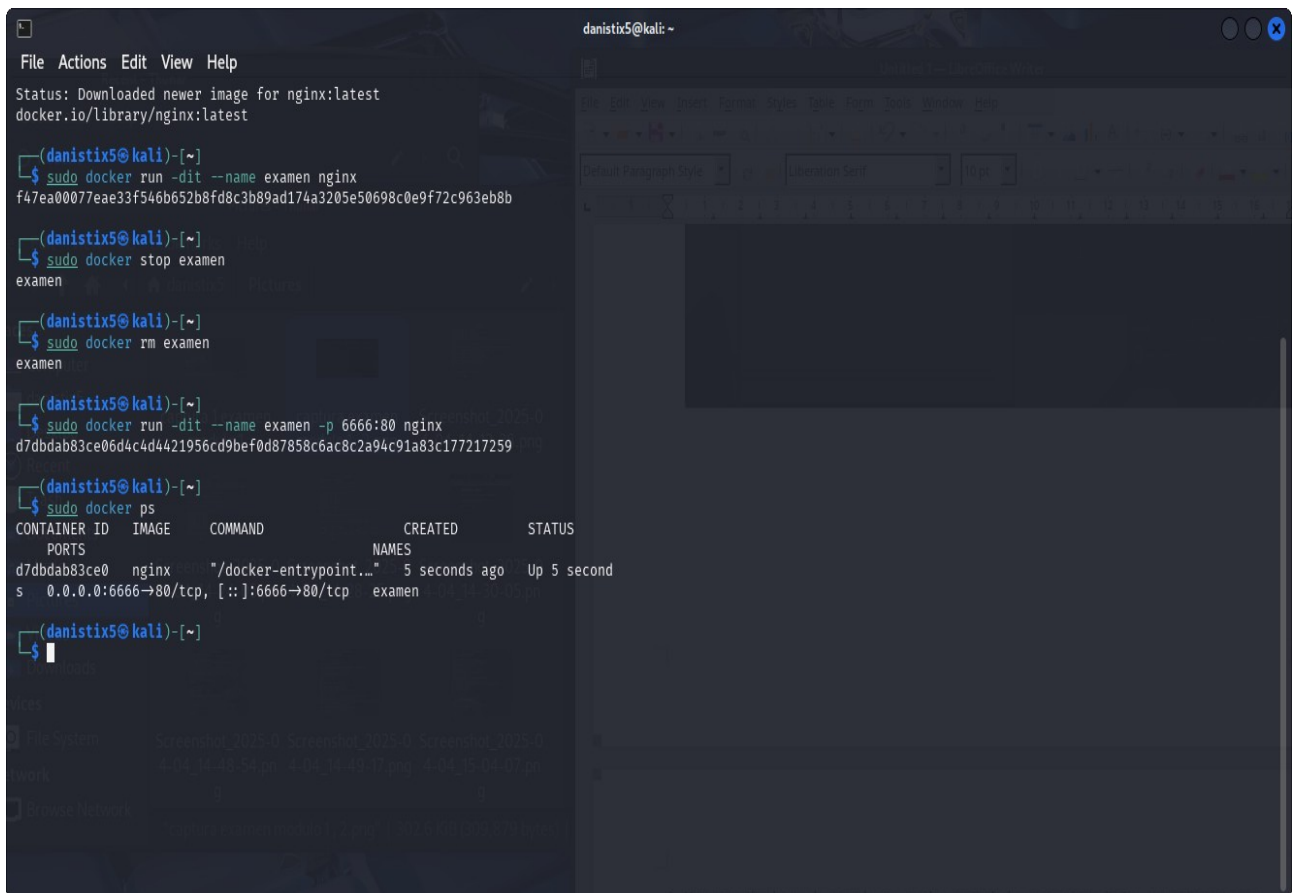
Apr 11 09:35:16 kali dockerd[771]: time="2025-04-11T09:35:16.736846538+02:00"
Apr 11 09:35:16 kali dockerd[771]: time="2025-04-11T09:35:16.758586814+02:00"
Apr 11 09:35:17 kali dockerd[771]: time="2025-04-11T09:35:17.696942312+02:00"
Apr 11 09:35:17 kali dockerd[771]: time="2025-04-11T09:35:17.697393565+02:00"
Apr 11 09:35:17 kali dockerd[771]: time="2025-04-11T09:35:17.758330130+02:00"
Apr 11 09:35:17 kali dockerd[771]: time="2025-04-11T09:35:17.759921382+02:00"
Apr 11 09:35:17 kali dockerd[771]: time="2025-04-11T09:35:17.836782734+02:00"
Apr 11 09:35:17 kali dockerd[771]: time="2025-04-11T09:35:17.873491425+02:00"
Apr 11 09:35:17 kali dockerd[771]: time="2025-04-11T09:35:17.873553541+02:00"
Apr 11 09:35:17 kali systemd[1]: Started docker.service - Docker Application
[lines 1-23/23 (END)] ... skipping ...
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset:
   Active: active (running) since Fri 2025-04-11 09:35:17 CEST; 32min ago
 Invocation: 7ee476b6145f462f9fa0b23851a595e2
 TriggeredBy: ● docker.socket
   Docs: https://docs.docker.com
 Main PID: 771 (dockerd)
   Tasks: 9
  Memory: 99M (peak: 101M)
    CPU: 964ms
   CGroup: /system.slice/docker.service
           └─771 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Apr 11 09:35:16 kali dockerd[771]: time="2025-04-11T09:35:16.736846538+02:00" level=info msg="[graphdriver] using prior storage driver: overlay2"
Apr 11 09:35:16 kali dockerd[771]: time="2025-04-11T09:35:16.758586814+02:00" level=info msg="Loading containers: start."
Apr 11 09:35:17 kali dockerd[771]: time="2025-04-11T09:35:17.696942312+02:00" level=warning msg="error locating sandbox id ae015d094066a9ad3d69abd81f8504c8c12eac: sandbox ae015d094066a9ad3d69abd81f8504c8c12eac"
Apr 11 09:35:17 kali dockerd[771]: time="2025-04-11T09:35:17.697393565+02:00" level=info msg="Loading containers: done."
Apr 11 09:35:17 kali dockerd[771]: time="2025-04-11T09:35:17.758330130+02:00" level=info msg="Docker daemon" commit=6430e49 containerd-snapshotter=false storage-driver=overlay2 version=28.0.4
Apr 11 09:35:17 kali dockerd[771]: time="2025-04-11T09:35:17.759921382+02:00" level=info msg="Initializing buildkit"
Apr 11 09:35:17 kali dockerd[771]: time="2025-04-11T09:35:17.836782734+02:00" level=info msg="Completed buildkit initialization"
Apr 11 09:35:17 kali dockerd[771]: time="2025-04-11T09:35:17.873491425+02:00" level=info msg="Daemon has completed initialization"
Apr 11 09:35:17 kali dockerd[771]: time="2025-04-11T09:35:17.873553541+02:00" level=info msg="API listen on /run/docker.sock"
```

2. Ahora, vamos a crear un contenedor (llamado examen) basado en la imagen de nginx.



3. Una vez verificado que el contenedor esta creado y corriendo, lo vamos a parar, eliminar y posteriormente crear correctamente expuesto al puerto 6666.



```
(danistix5@kali)~$ sudo docker run -dit --name examen nginx
f47ea00077eae33f546b652b8fd8c3b89ad174a3205e50698c0e9f72c963eb8b

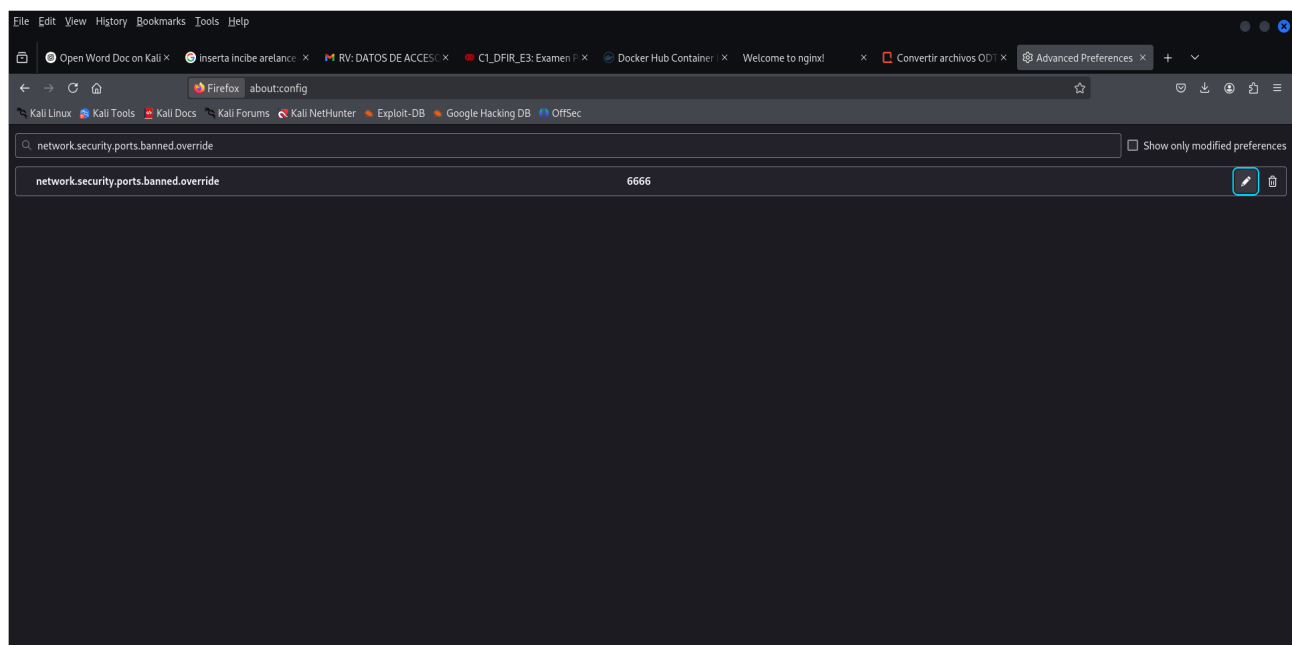
(danistix5@kali)~$ sudo docker stop examen
examen

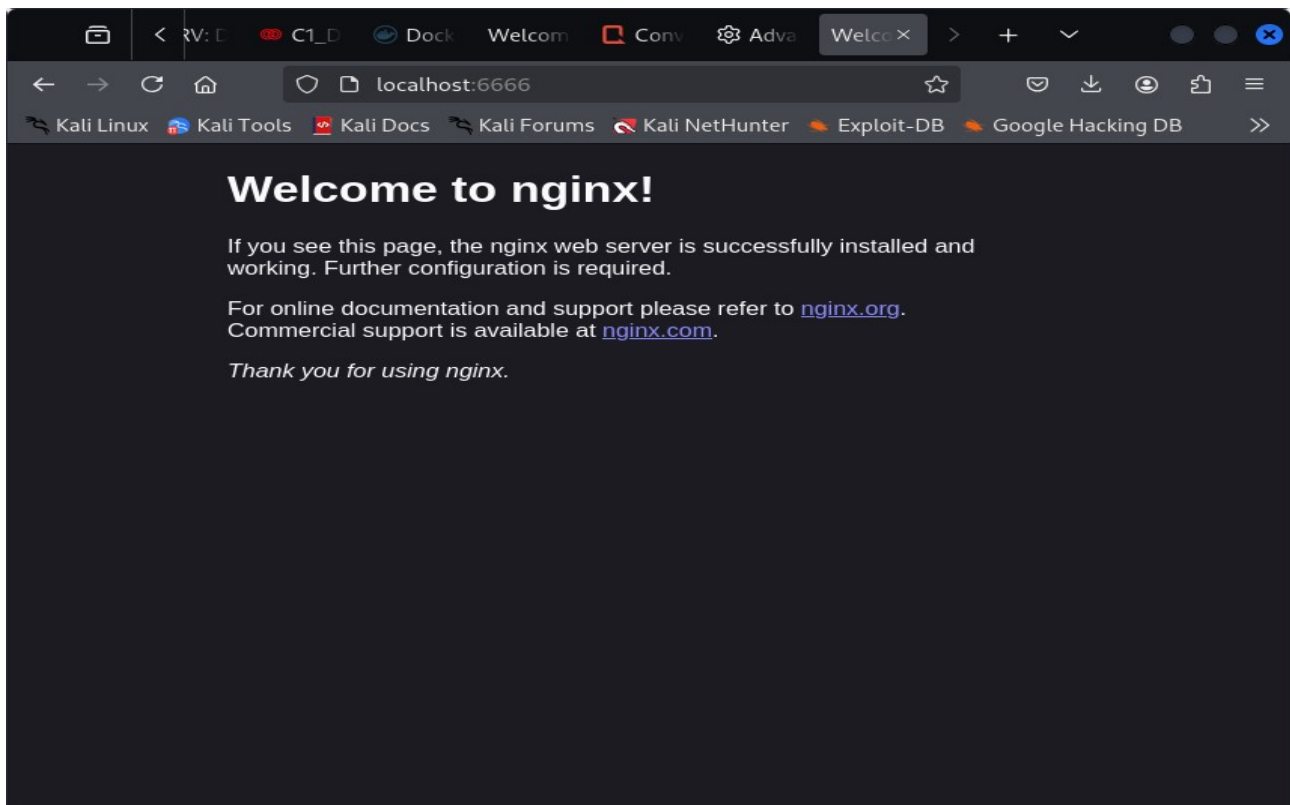
(danistix5@kali)~$ sudo docker rm examen
examen

(danistix5@kali)~$ sudo docker run -dit --name examen -p 6666:80 nginx
d7dbdab83ce06d4c4d4421956cd9bef0d87858c6ac8c2a94c91a83c177217259

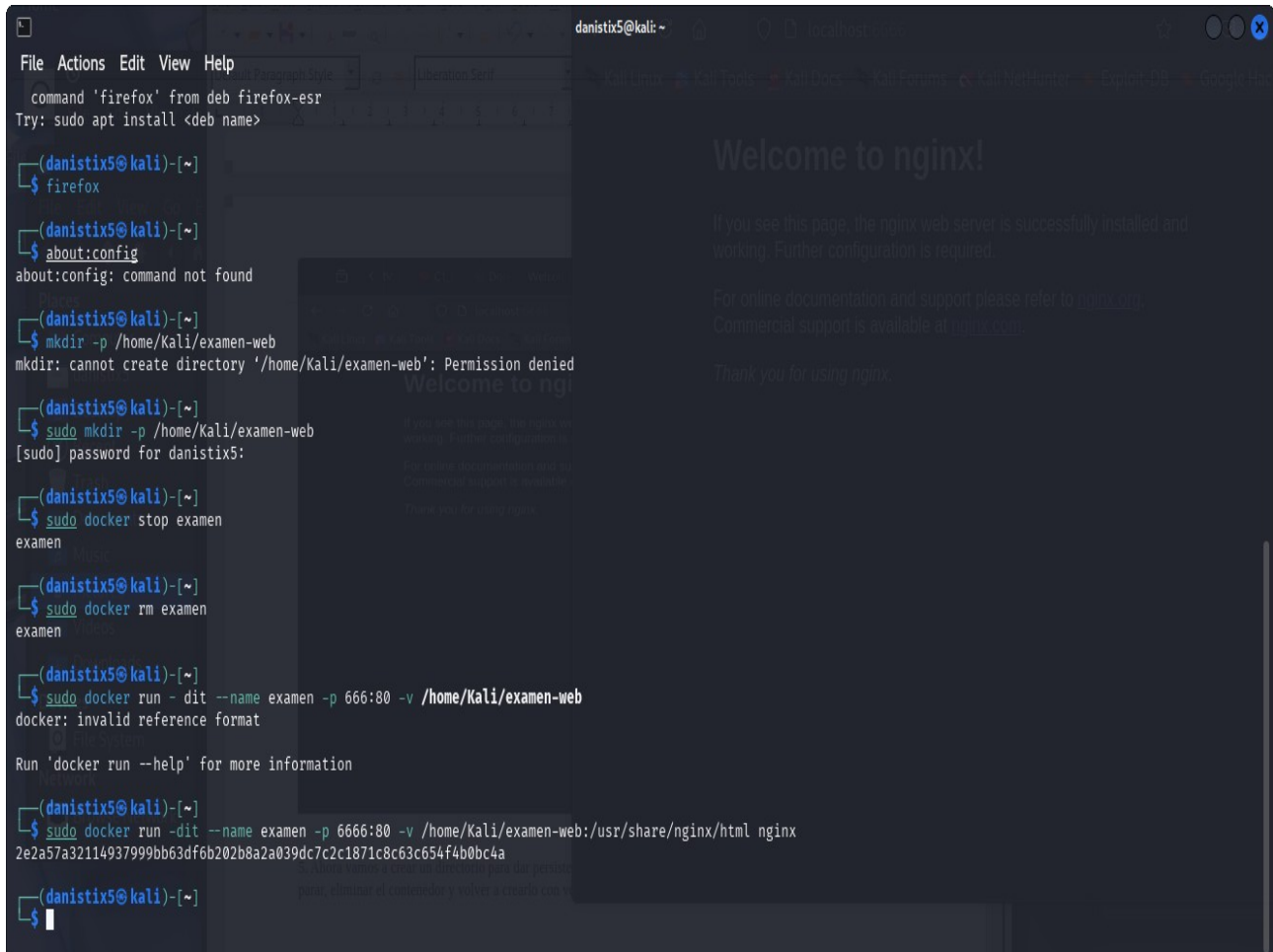
(danistix5@kali)~$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS
PORTS         NAMES
d7dbdab83ce0   nginx    "/docker-entrypoint..." 5 seconds ago  Up 5 second
s             0.0.0.0:6666->80/tcp, [::]:6666->80/tcp   examen
```

4. Vamos a conectarnos al servidor web y comprobar que funciona. Vemos que nuestro navegador (Firefox) bloquea por defecto ciertos puertos (incluyendo el 6666) al considerarlo no seguro. Habilitamos el puerto en nuestro navegador.





5. Ahora vamos a crear un directorio para dar persistencia de datos y hacerlo accesible al contenedor. Tendremos que parar, eliminar el contenedor y volver a crearlo con volumen persistente.



6. Finalmente, vamos a escribir un archivo HTML en el directorio compartido y visualizarlo en el navegador. Lo hacemos en HTML que es lo mas adecuado en un servidor web.

```
danistix5@kali: ~  
$ firefox  
(danistix5@kali)-[~]  
$ about:config  
about:config: command not found  
(danistix5@kali)-[~]  
$ mkdir -p /home/Kali/examen-web  
mkdir: cannot create directory '/home/Kali/examen-web': Permission denied  
(danistix5@kali)-[~]  
$ sudo mkdir -p /home/Kali/examen-web  
[sudo] password for danistix5:  
(danistix5@kali)-[~]  
$ sudo docker stop examen  
examen  
(danistix5@kali)-[~]  
$ sudo docker rm examen  
examen  
(danistix5@kali)-[~]  
$ sudo docker run -dit --name examen -p 6666:80 -v /home/Kali/examen-web  
docker: invalid reference format  
Run 'docker run --help' for more information  
(danistix5@kali)-[~]  
$ sudo docker run -dit --name examen -p 6666:80 -v /home/Kali/examen-web:/usr/share/nginx/html nginx  
2e2a57a32114937999bb63df6b202b8a2a039dc7c2c1871c8c63c654f4b0bc4a  
(danistix5@kali)-[~]  
$ echo "<h1>Examen Docker Daniel Stix</h1>" | sudo tee /home/Kali/examen-web/index.html  
<h1>Examen Docker Daniel Stix</h1>  
(danistix5@kali)-[~]  
$
```

