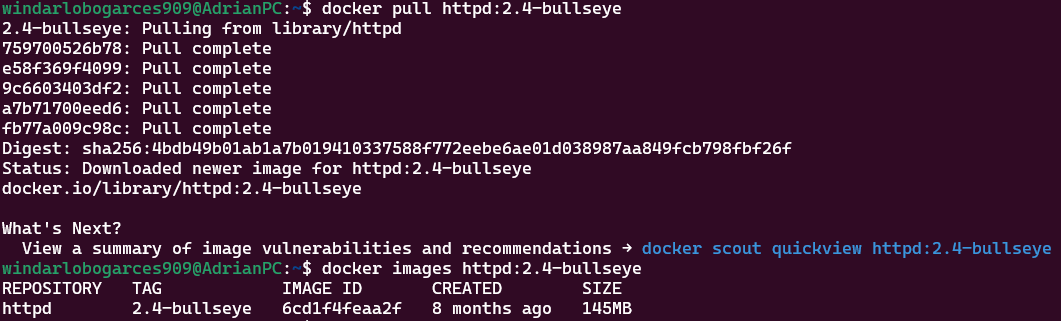
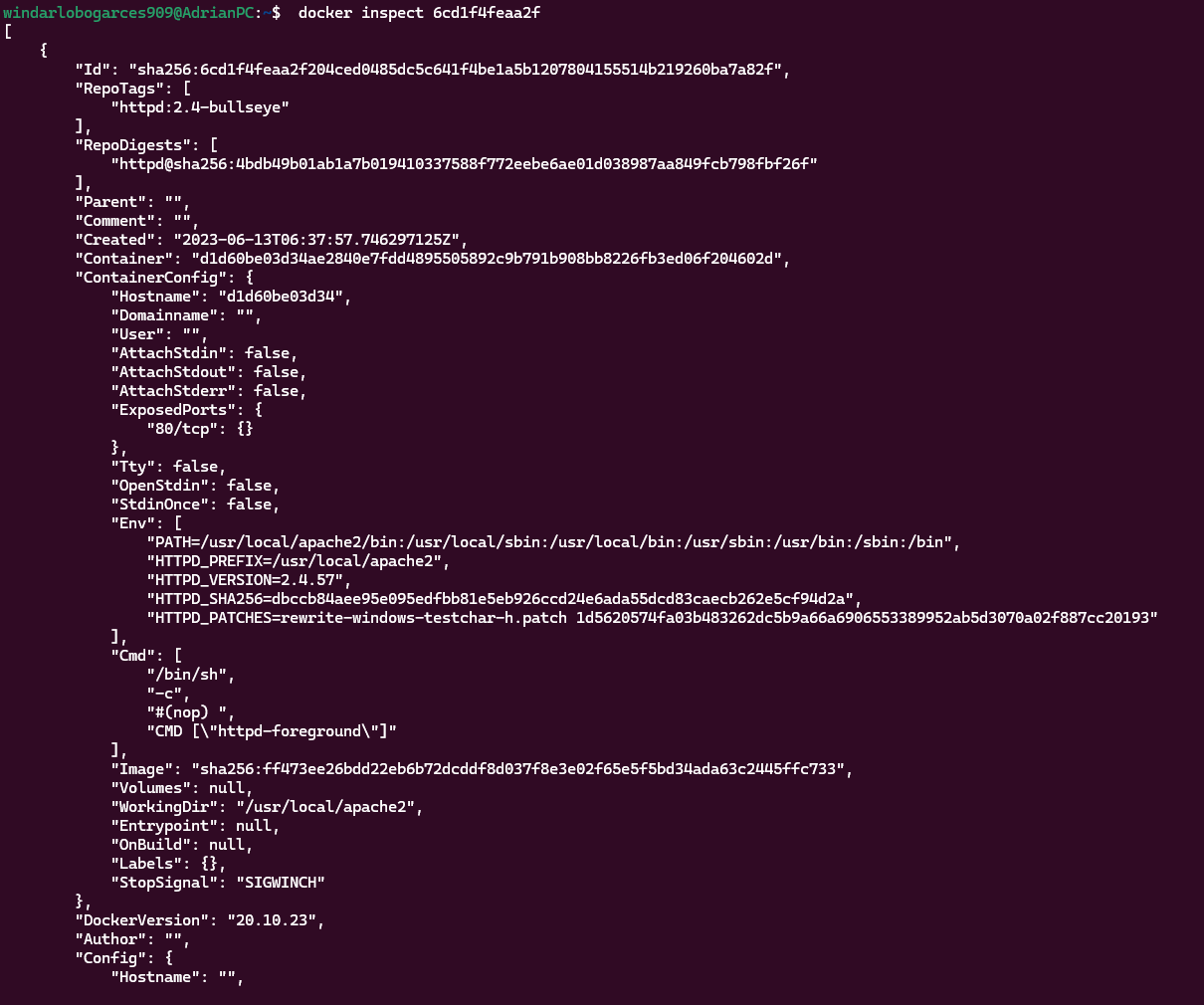
**Tarea 1**

DAW - 23/24

* **Docker. Imágenes**

1. **El tamaño que ocupa la imagen servidor web apache HTTP en su versión 2.4 bullseye en disco**
   1. 145MB.
2. **¿Cuál es el ID de la imagen que acabas de descargar?¿Y su hash o**

**Digest? ¿Qué relación tienen entre sí?**





* 1. **Digest:** httpd@sha256:4bdb49b01ab1a7b019410337588f772eebe6ae01d038987aa849fcb798fbf26f.

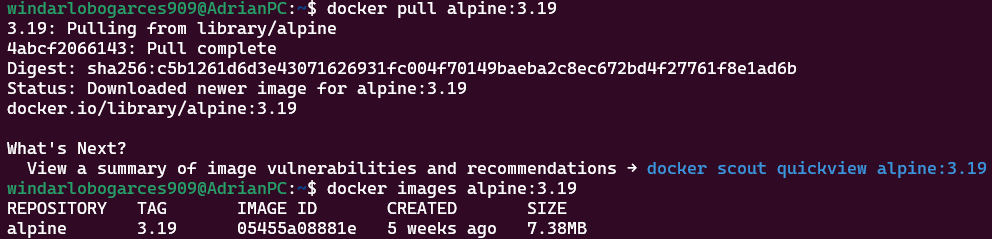
La cadena de hash es única de la imagen en el registro de Docker

* 1. **El ID de la imagen:**

feaa2f204ced0485dc5c641f4be1a5b1207804155514b219260ba7a82f. El ID de la imagen es identificador único asignado por Docker

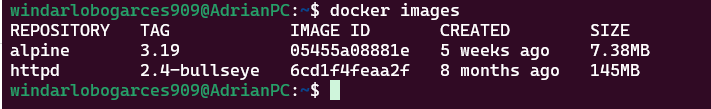
* 1. El digest se calcula basándose en el contenido único de la imagen y puede utilizarse para asegurar su integridad, lo que significa que cualquier cambio en el contenido generará un digest diferente. Por otro lado, el ID de la imagen es un identificador asignado internamente por Docker para su gestión operativa. Aunque son diferentes, el digest y el ID de la imagen están relacionados en el sentido de que ambos son importantes para el funcionamiento y la gestión adecuada de las imágenes Docker.

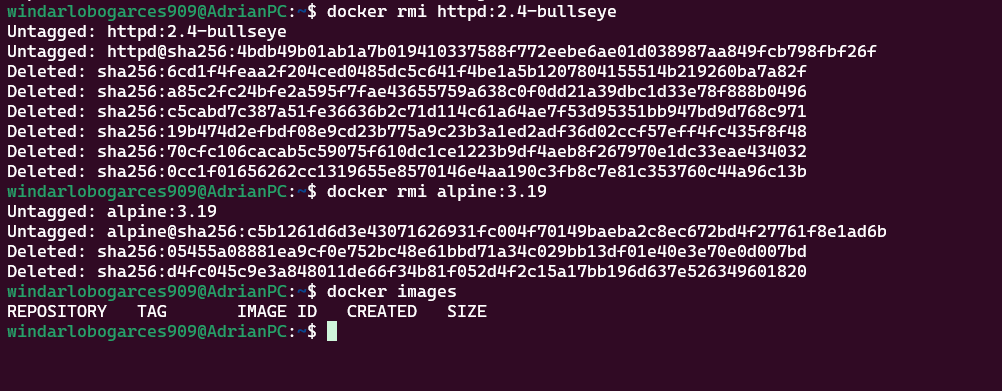
1. **Descarga ahora la versión 3.19 alpine de la imagen ¿Cuánto ocupa en disco? ¿Cuántas veces más pequeña o grande es la imagen de alpine respecto de la de bullseye? ¿A qué se debe esta diferencia?**
   1. 7.38MB



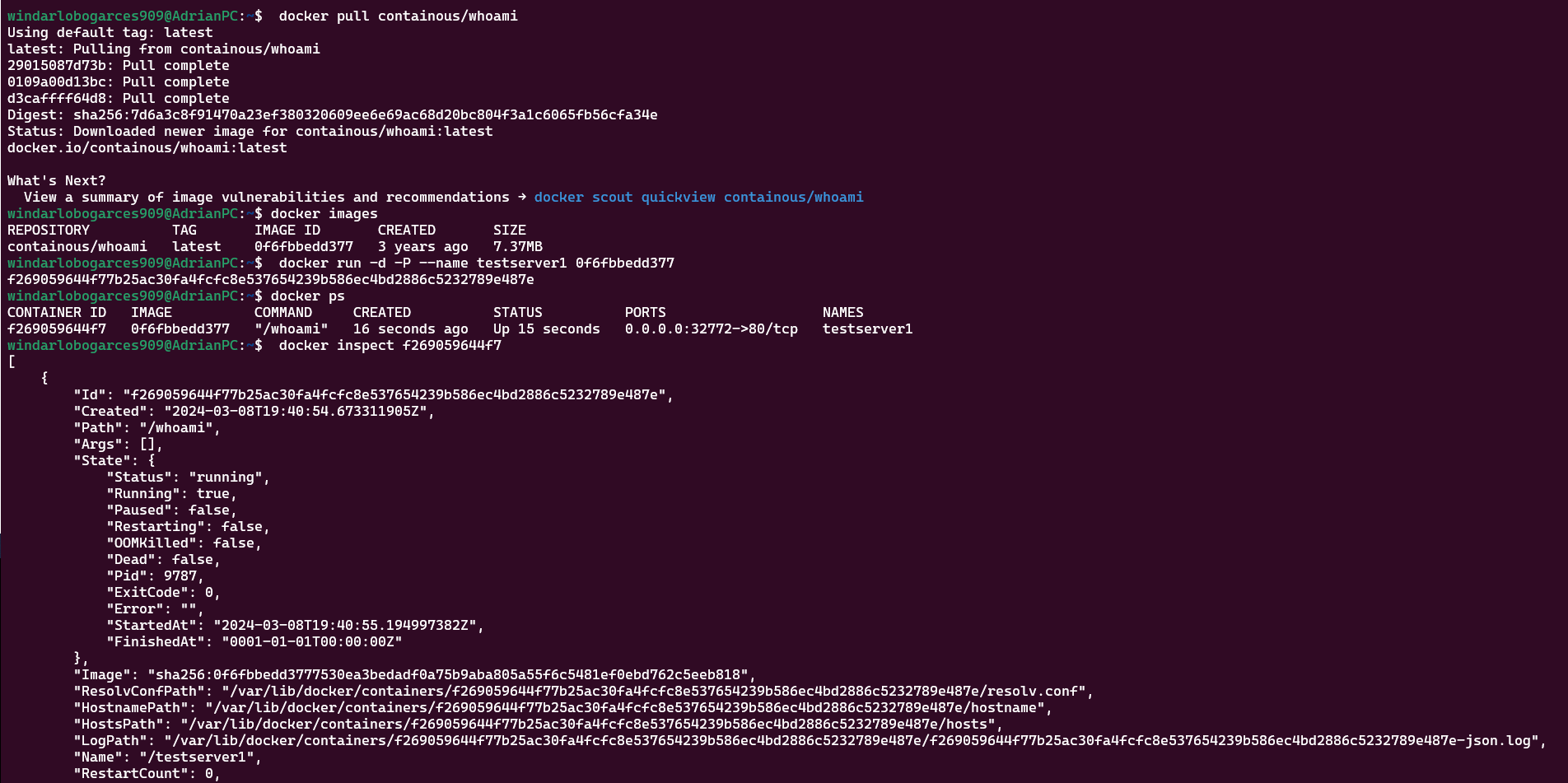
* 1. La variación en el tamaño entre Alpine y Debian (Bullseye) se explica por varios factores. Esto incluye el diseño general de la distribución, la selección de paquetes, la distribución base y el gestor de paquetes utilizado en cada sistema. Alpine opta por musl libc y BusyBox, adoptando un enfoque minimalista en la selección de paquetes. Por otro lado, Debian (Bullseye) emplea glibc y ofrece un amplio conjunto de paquetes y herramientas para satisfacer una variedad de necesidades.
  2. Alpine se concentra en ser eficiente y minimizar el uso de recursos. Por eso, suele ser más ligero y consumir menos recursos en comparación con Debian (Bullseye). Mientras tanto, Debian (Bullseye) está diseñado para ser versátil y ofrecer una amplia variedad de herramientas y paquetes para satisfacer diferentes necesidades.

1. **Elimina ambas imágenes del registro local**

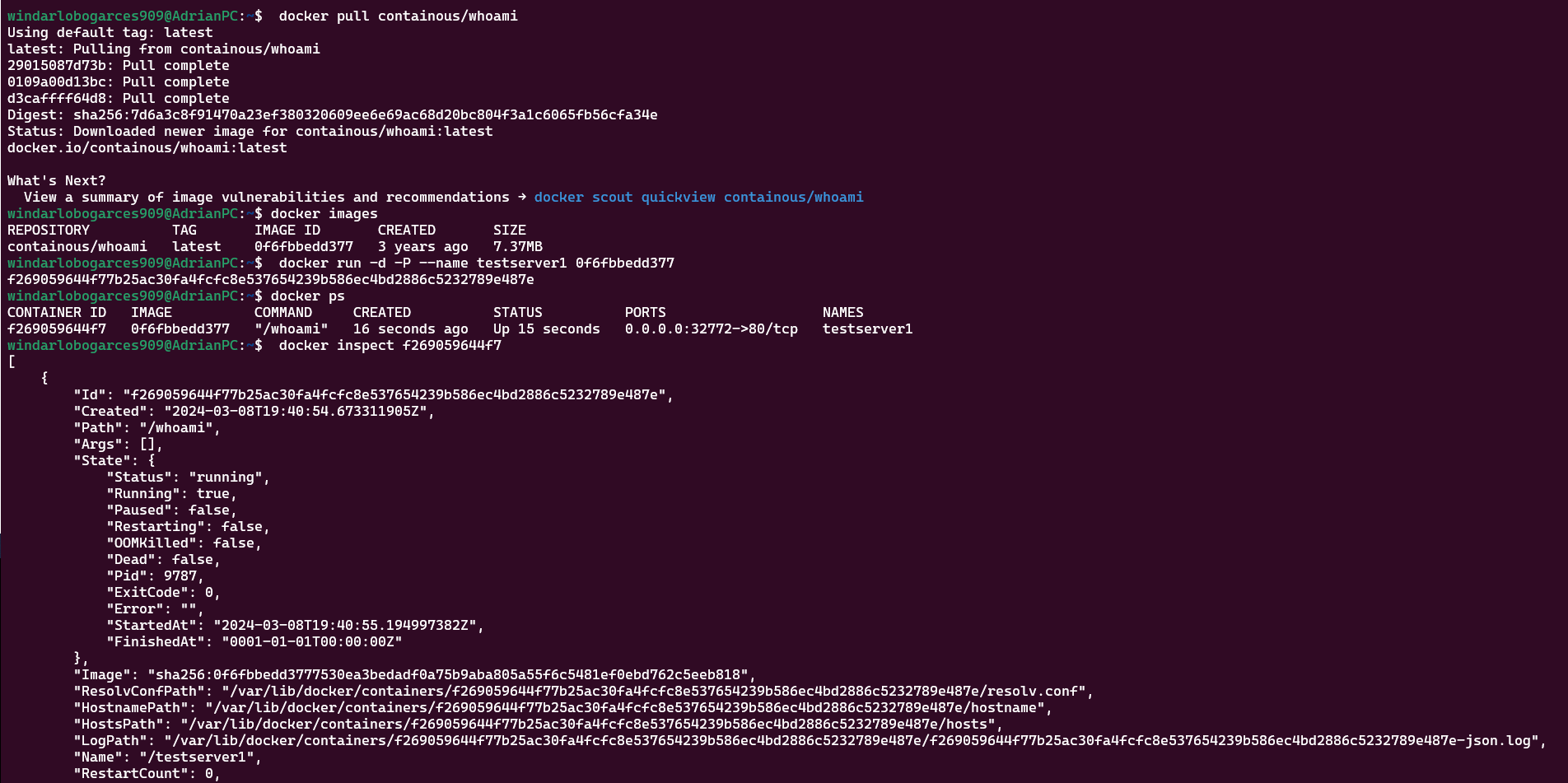




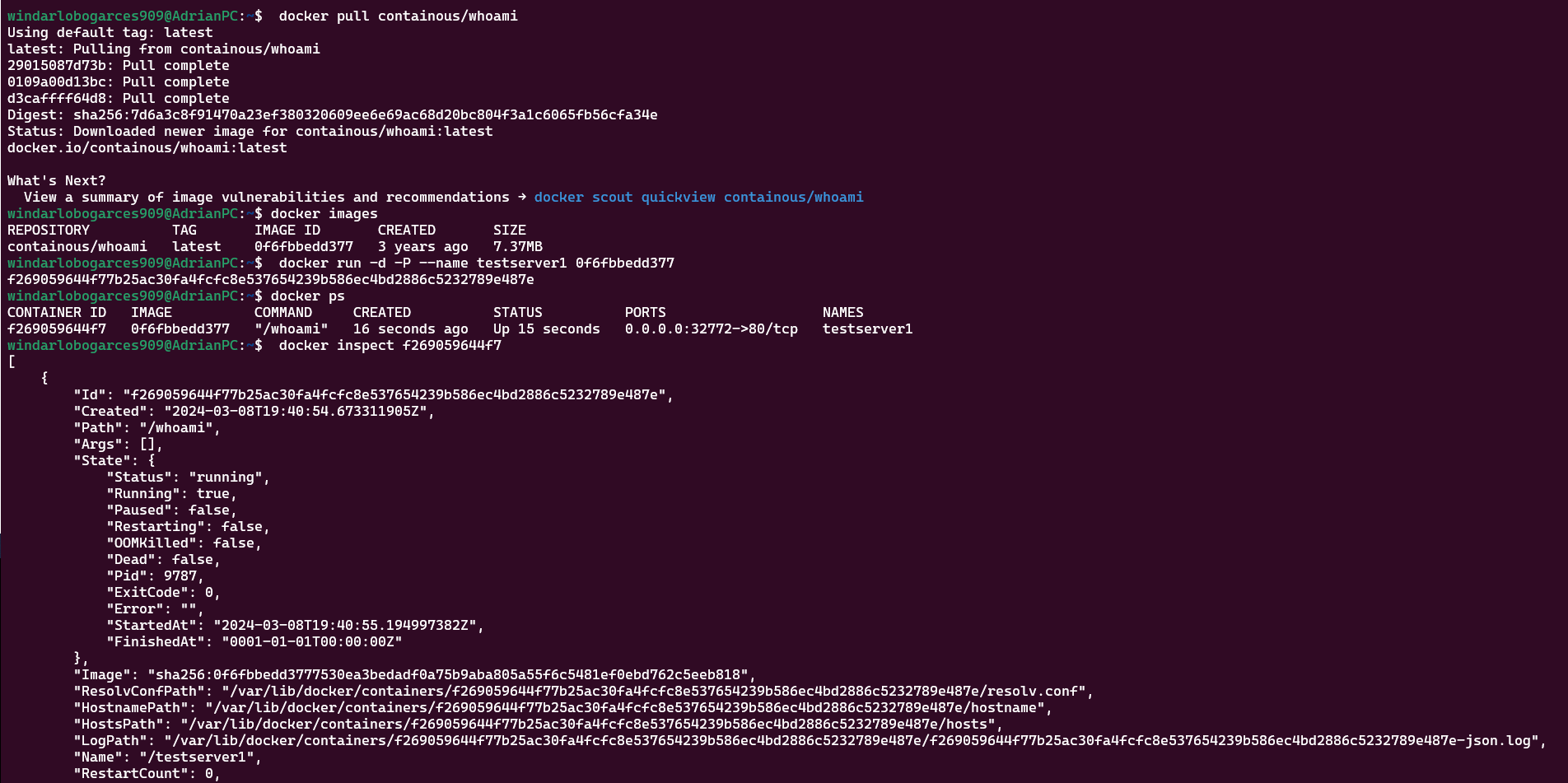
* **Docker. Containers y redes**

1. **Descarga la última versión de la imagen containous/whoami**.
2. **Levanta un contenedor llamado testserver1 a partir de la imagen anterior**

**asignando un puerto aleatorio del sistema host (opción -P). Ejecútalo en segundo plano.**



1. **Inspecciona el contenedor anterior para determinar el puerto del host que**

**se ha asignado. Accede a la URL http://localhost:PUERTO desde tu navegador**





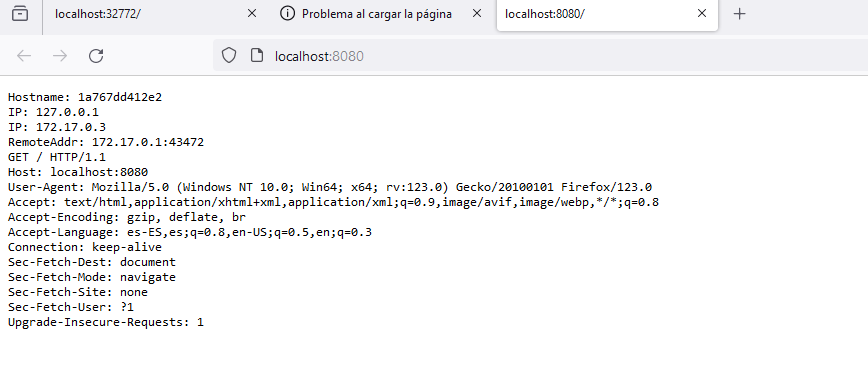


1. **Crea un contenedor idéntico al anterior con el nombre testserver2, pero**

**en esta ocasión enlaza los puertos 8080 y 4433 del host a los puertos 80 y 443 del contenedor .**

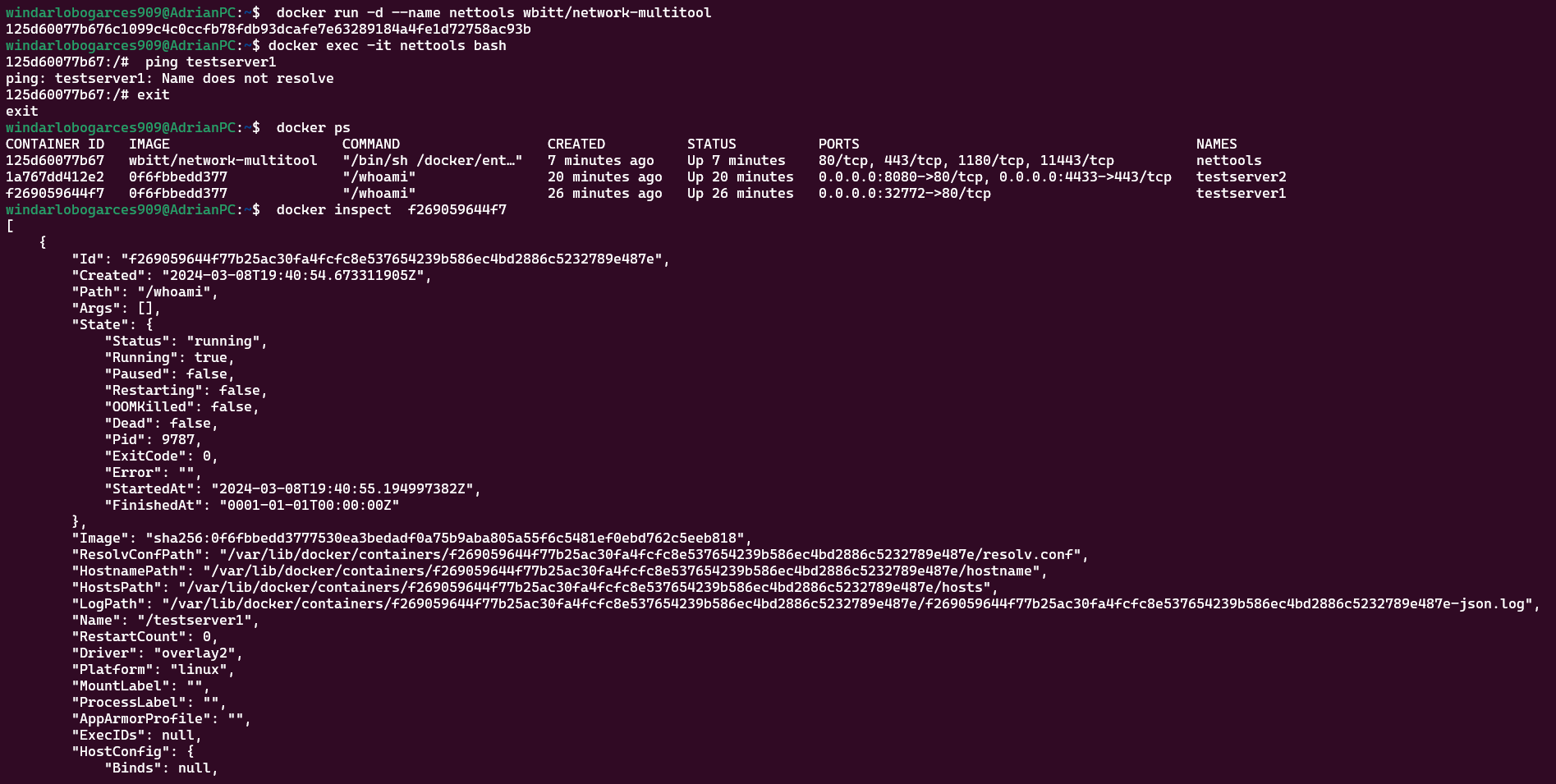
1. Accede a las URLS http://localhost:8080 y https://localhost:4433 desde tu navegador

El puerto 443 esta en conflicto con el puerto 80, ambos puertos son estándar utilizados para el tráfico HTTP. Por lo general el puerto 80 se utiliza para HTTP sin cifrar, mientras que el puerto 443 se utiliza para HTTPS (Hyper Text Transfer Protocol Secure), que es HTTP seguro (van cifrados los datos). En si la imagen Whoami, solo funciona con conexiones HTTP.

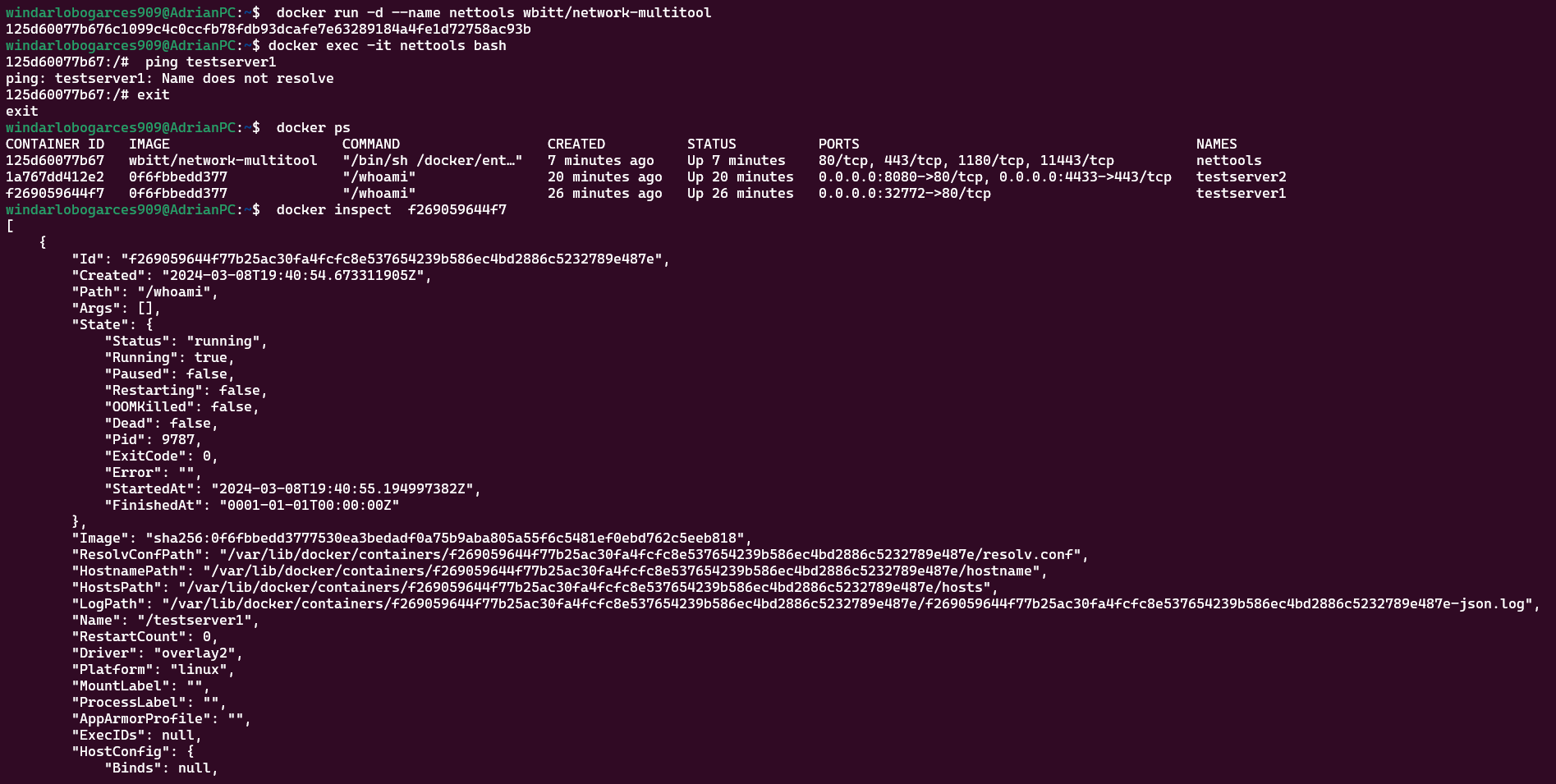


1. **Levanta un contenedor a partir de la imagen wbitt/network-multitool con**

**el nombre nettools. No es necesario que publiques ningún puerto en el host**



1. **Utilizando el comando docker exec -it nettools bash accede al**

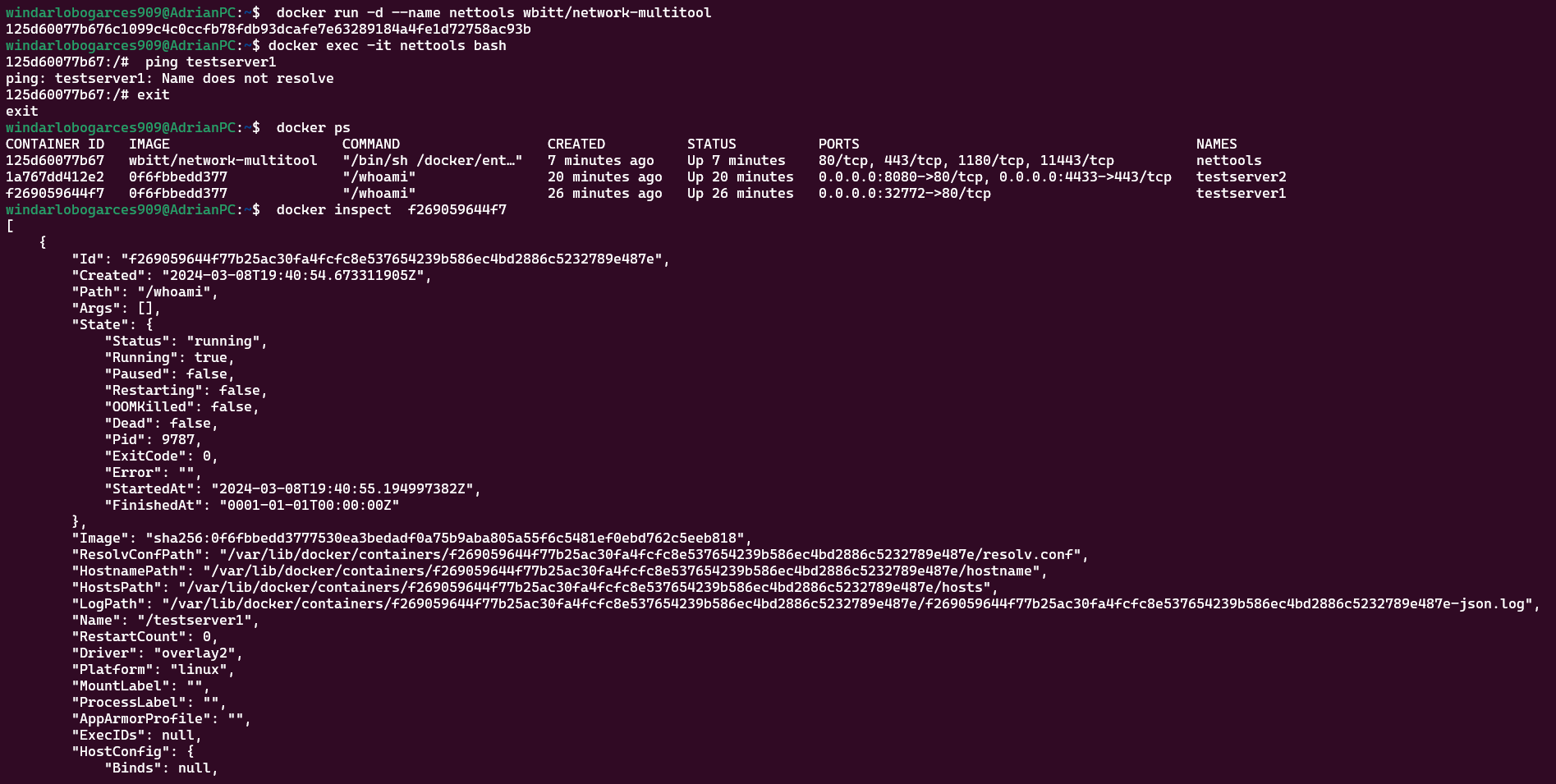
**contenedor nettools y ejecuta el comando ping testserver1 ¿Obtienes respuesta? En caso negativo, ¿por qué no?**

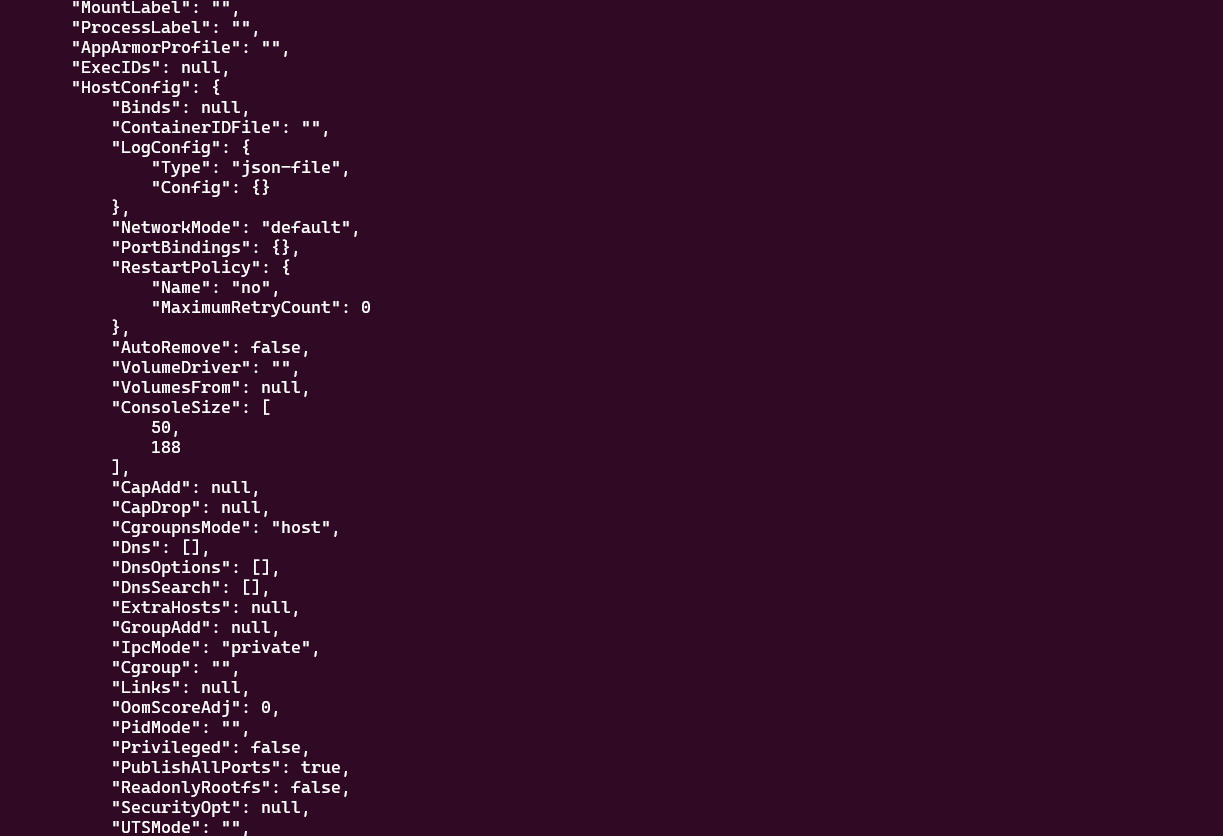
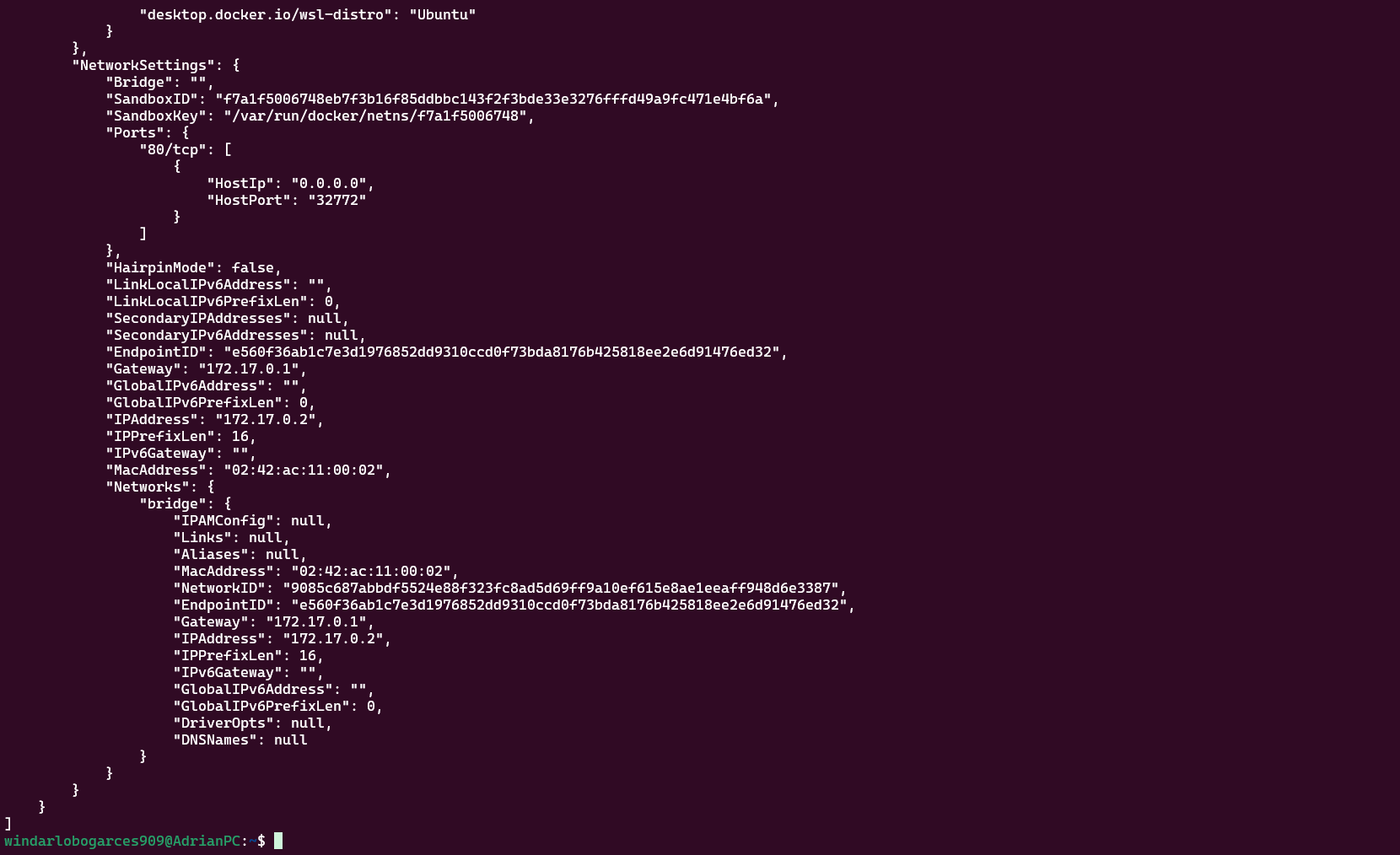
* 1. No, no obtengo respuesta, al no estar en la misma red.
  2. el contenedor testserver1 no tiene un servidor DNS configurado o no tiene una entrada de hosts que asocie el nombre "testserver1" con su dirección IP.

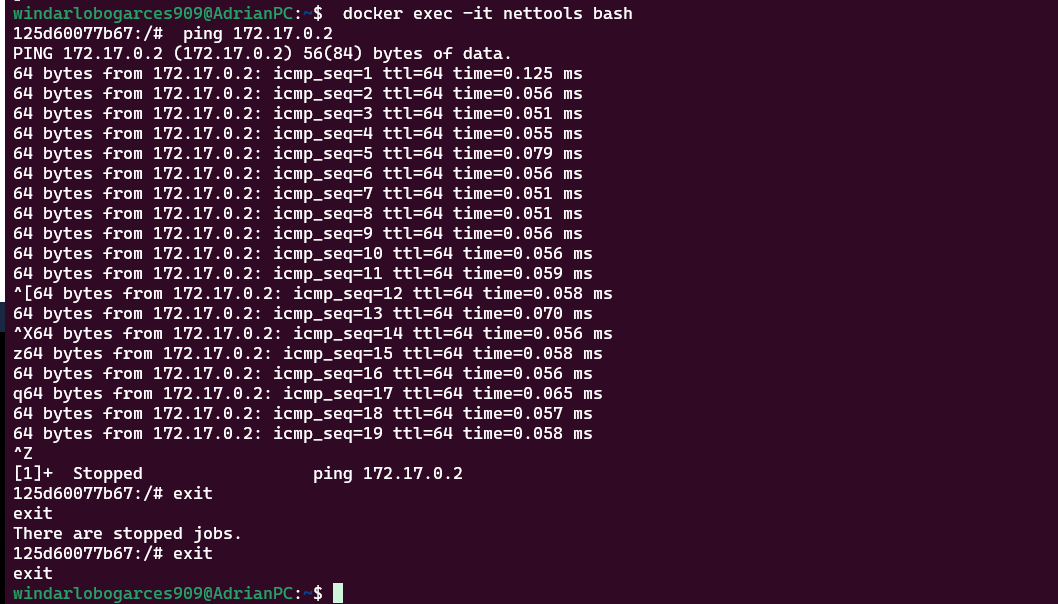
1. **Repite el punto anterior pero en esta ocasión haz ping a la IP del**

**contenedor testserver1 (tendrás que utilizar el comando docker inspect para**

**conocer la IP) ¿Obtienes respuesta? ¿Por qué?**



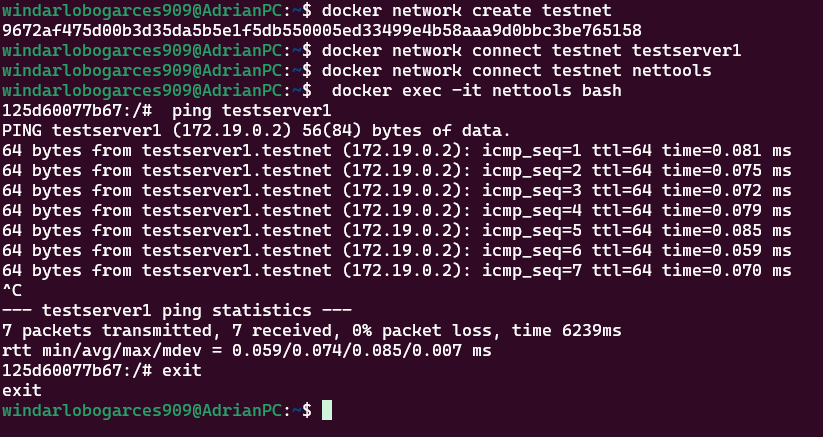


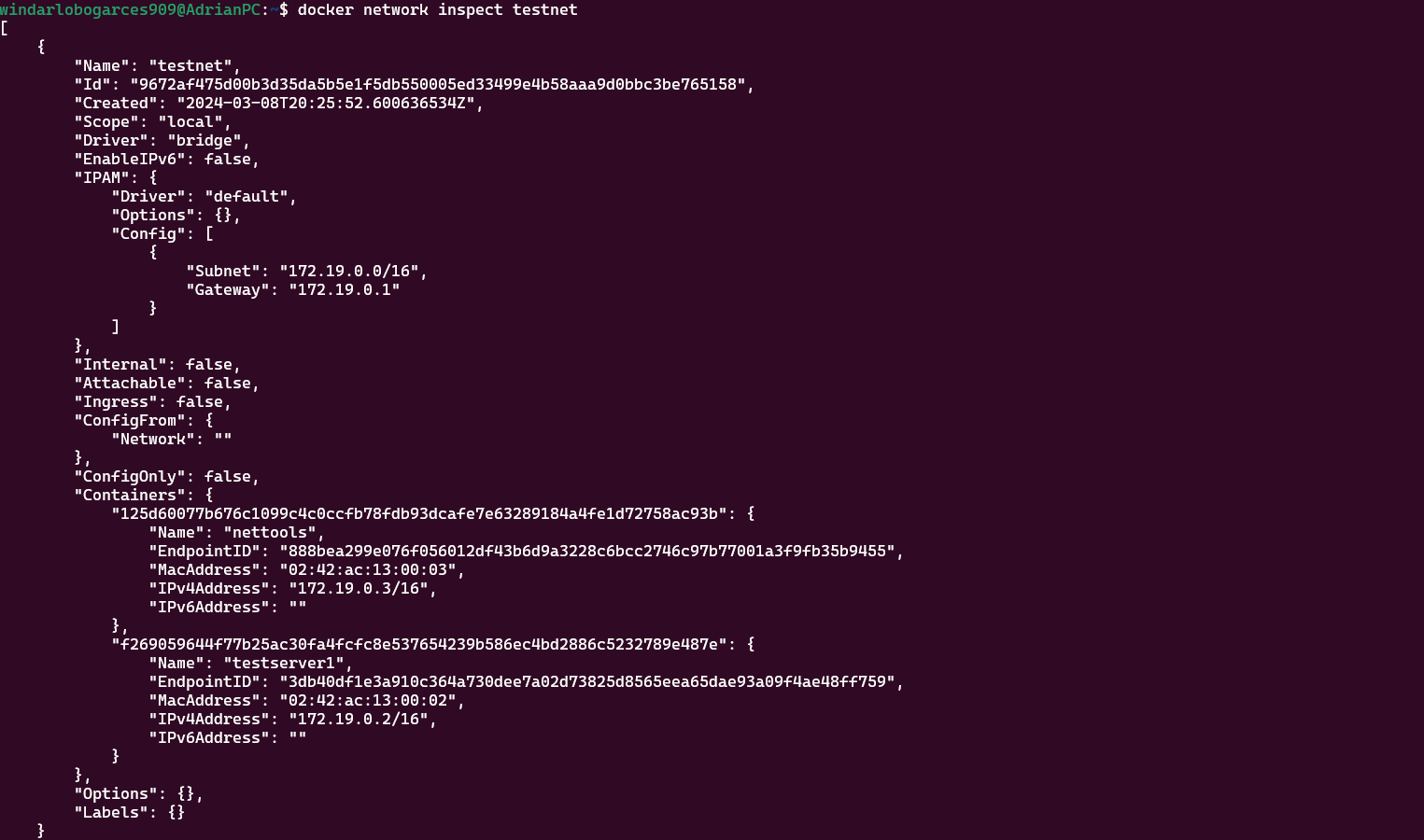


* 1. Si obtengo respuesta ya que la comunicación entre los contenedores nettools y testserver1 se establece correctamente debido a que ambos contenedores están en la misma red de puente de Docker.
  2. Porque cuando realizo el ping a la dirección IP del contenedor testserver1 desde el contenedor nettools, la comunicación se realiza a través de la red de puente de Docker, lo que permite que los contenedores se comuniquen entre sí.

1. **Crea una red llamada testnet y une los contenedores testserver1 y**

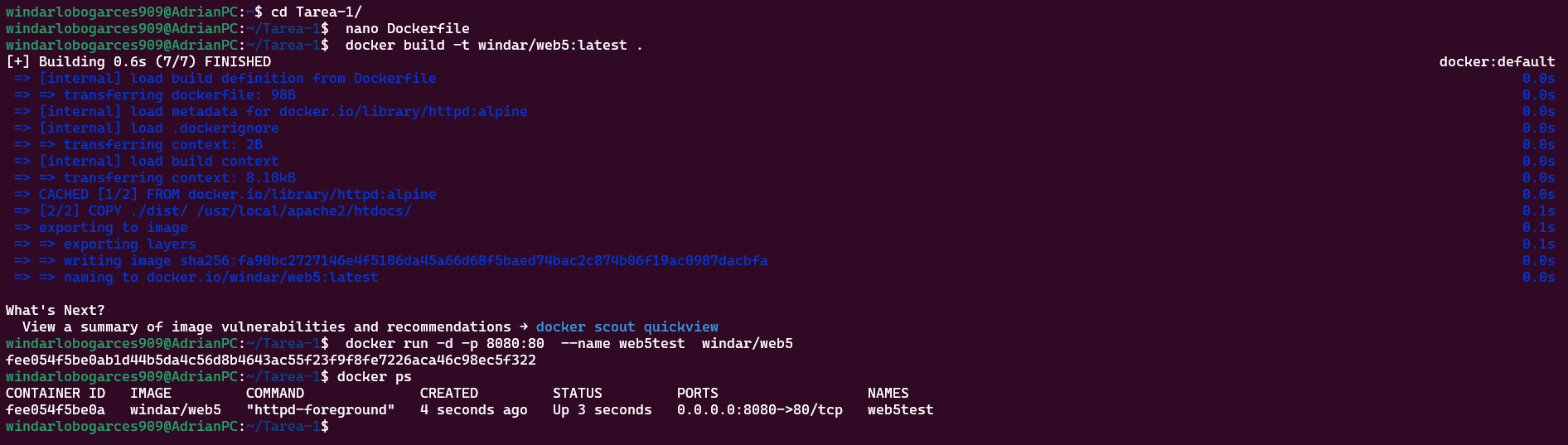
**nettools a la misma. Repite el apartado 6. Explica qué ocurre en esta ocasión**

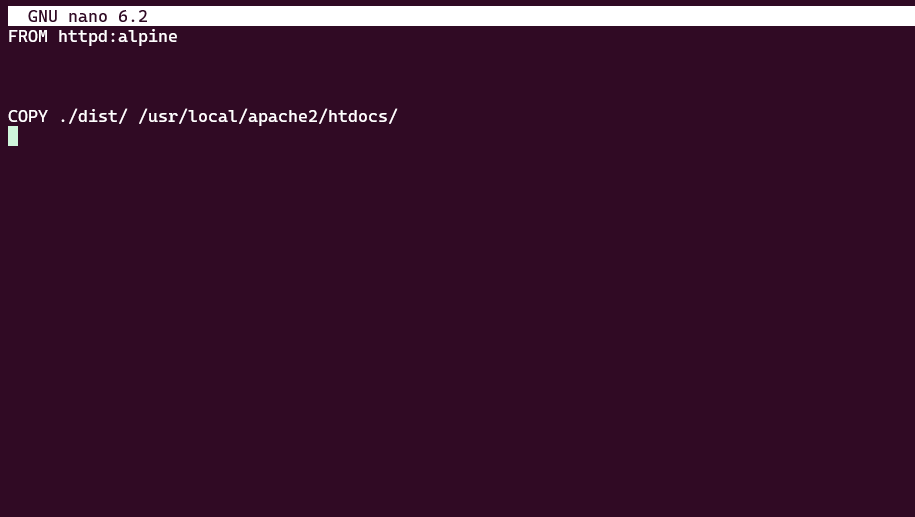




* 1. Ocurre que Docker resuelve el nombre testserver1 a la dirección IP 172.19.0.2, que es la dirección IP asignada al contenedor testserver1 en la red testnet. Luego, el ping se realiza a esta dirección IP, lo que resulta en respuestas exitosas del contenedor testserver1. Al conectar los contenedores a la misma red, se crea un entorno de red privada donde los contenedores pueden comunicarse de manera eficiente y segura.
* **Docker. Dockerfile y volúmenes**

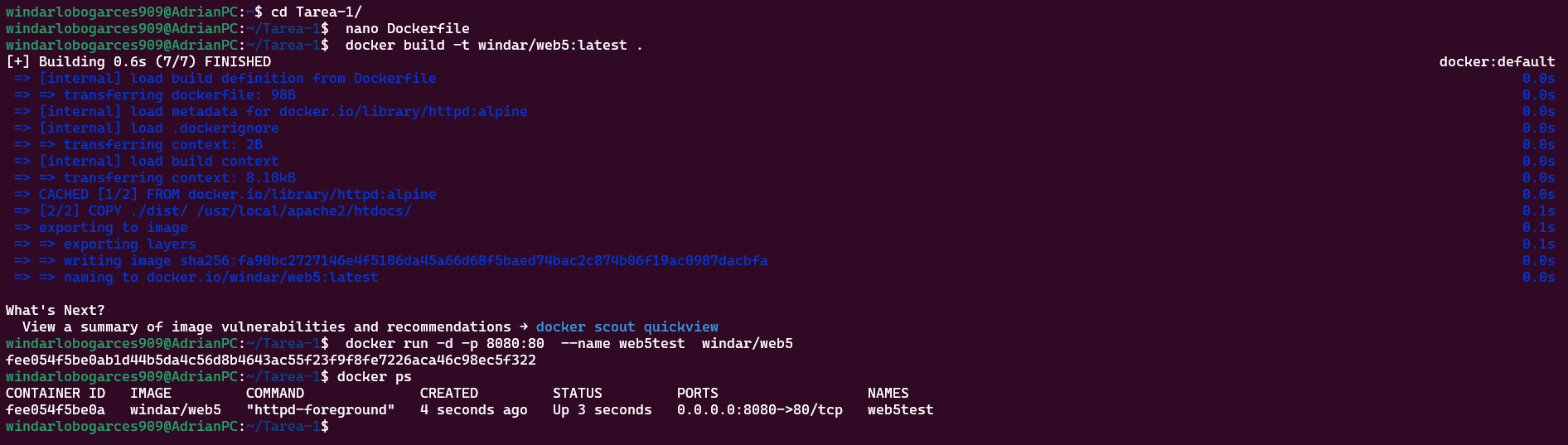
1. **Crea un Dockerfile que basado en la última versión de alpine de httpd que sirva una página web de muestra (no importa el contenido)**



****

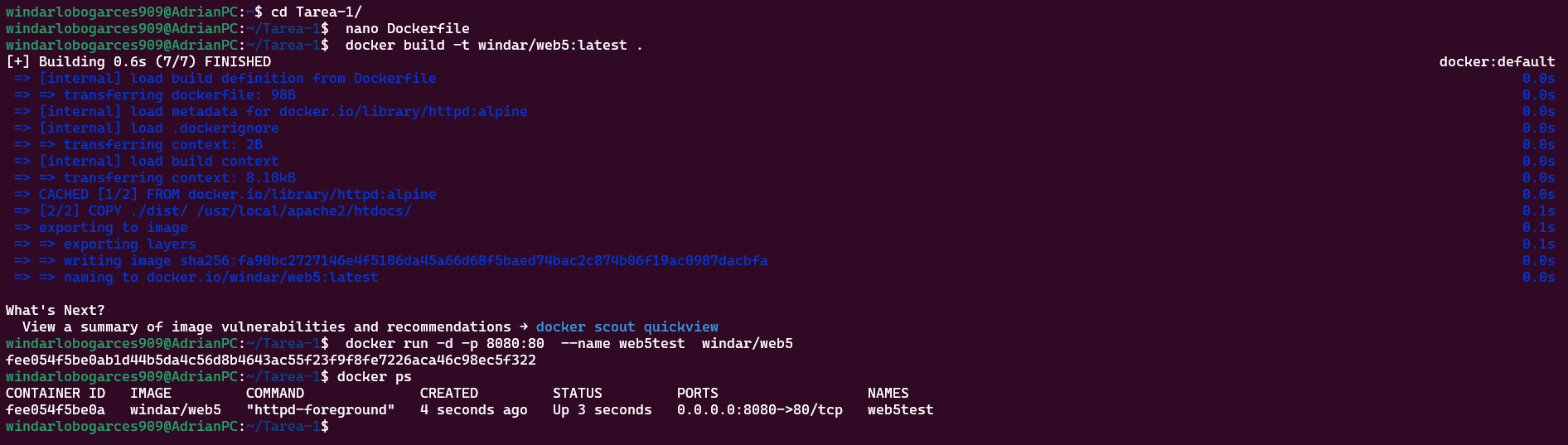
1. **Construye la imagen a partir del Dockerfile anterior con el tag**

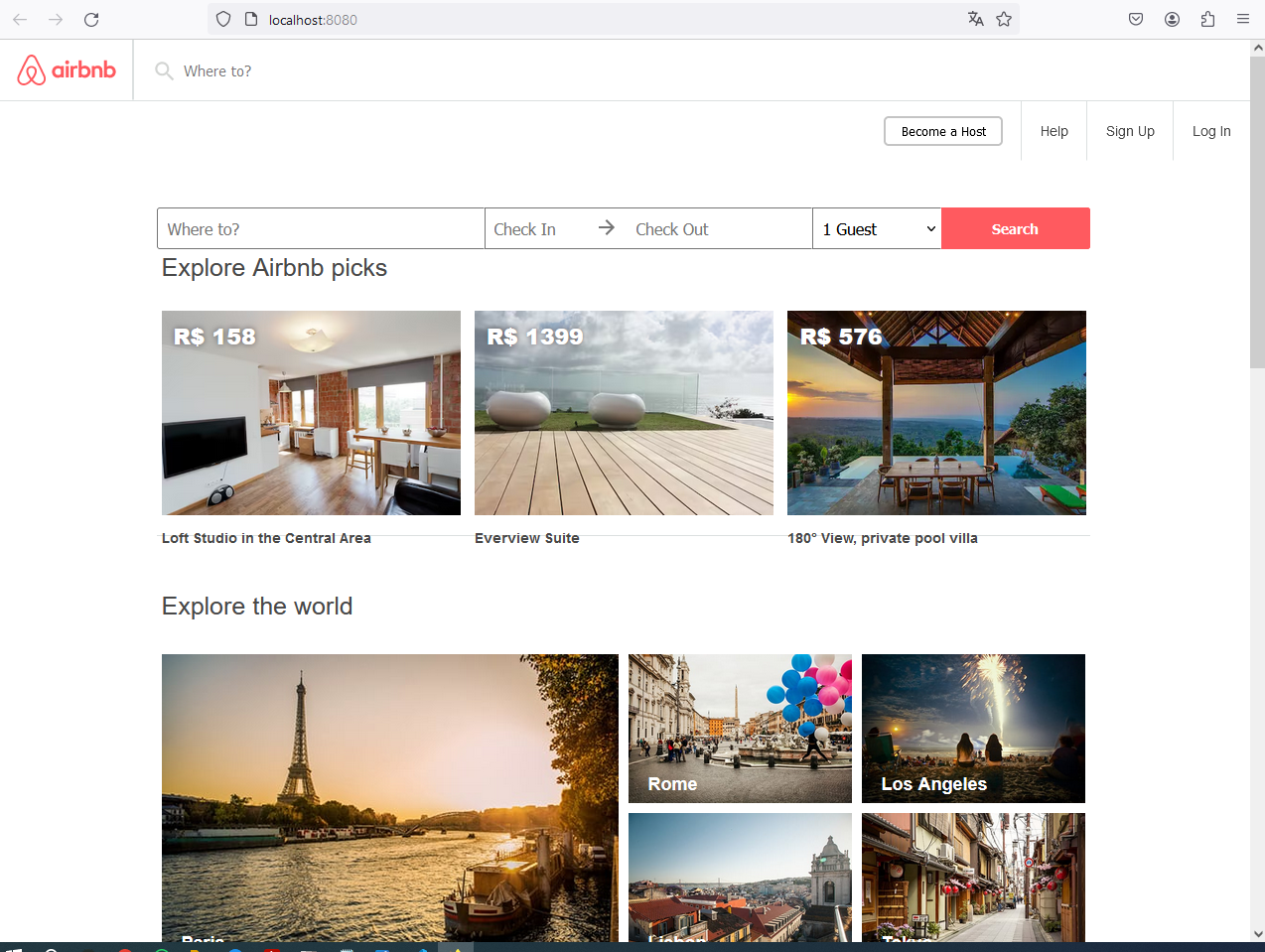
**{tunombre}/web5:latest**

****

1. **Levanta un contenedor de la imagen anterior llamado web5test que**

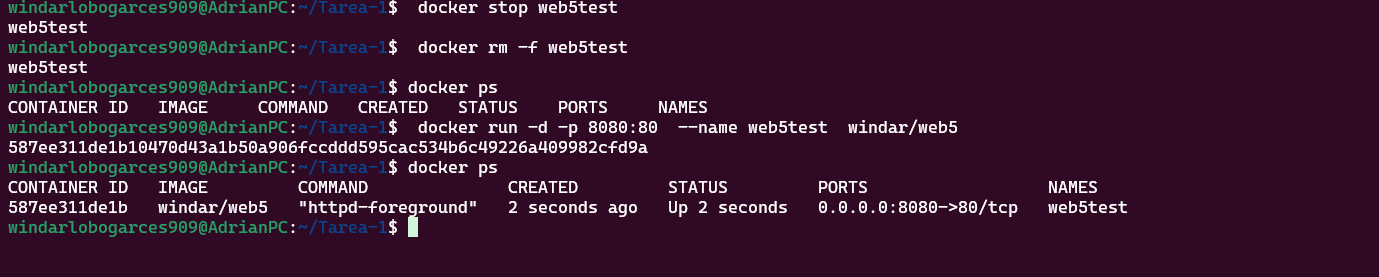
**publique el puerto 80 en el 8080 del host. ¿Puedes acceder a la web enhttp://localhost:8080 ? En caso negativo ¿Qué está sucediendo?**

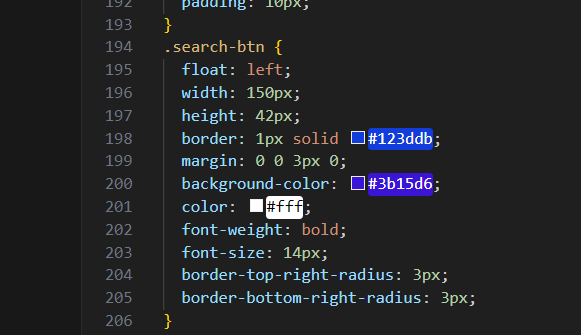
**¿Cómo lo arreglarías?**

* 1. Si puedo acceder a la pagina web

1. **Detén el contenedor y elimínalo. Haz un cambio cualquiera en el fichero**

**html de la web y vuelve a levantarlo (sin reconstruir la imagen). ¿Se aplicaron los cambios en la web? En caso negativo ¿qué ha sucedido?**

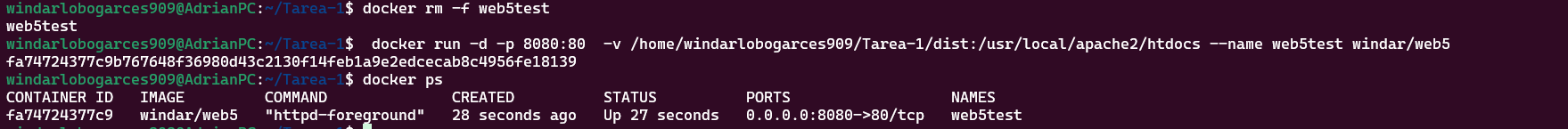
****

* 1. Realizo el cambio en el .search-btn

* 1. No se aplicaron los cambios, Docker utiliza una copia del sistema de archivos del contenedor al iniciar el contenedor descarta todos los cambios realizados dentro del contenedor cuando se detiene o elimina.
  2. Sucede que si deseo mostrar los cambios es necesario utilizar volúmenes para montar directorios del host dentro del contenedor.

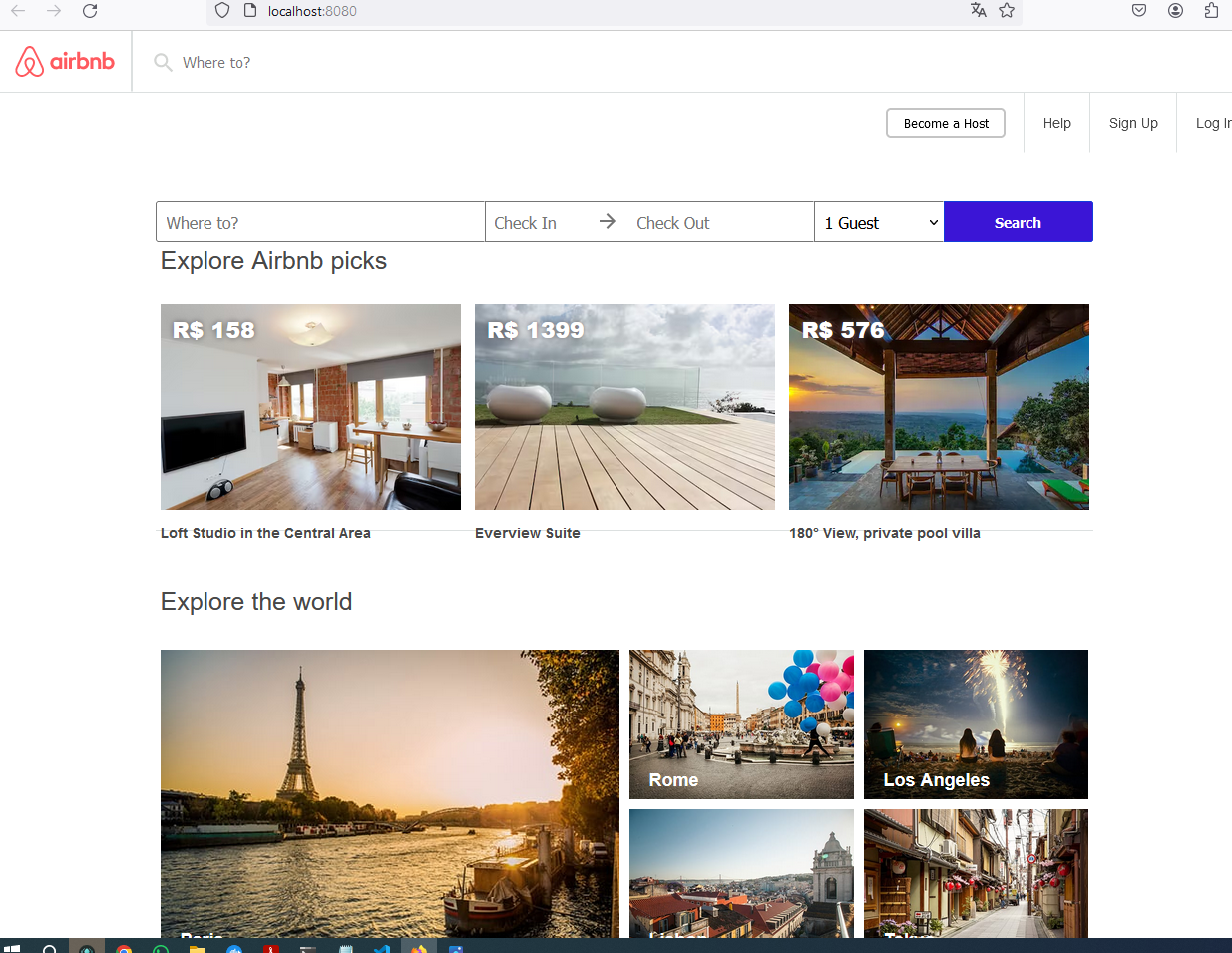
1. **A continuación levanta un nuevo contenedor con el mismo nombre**

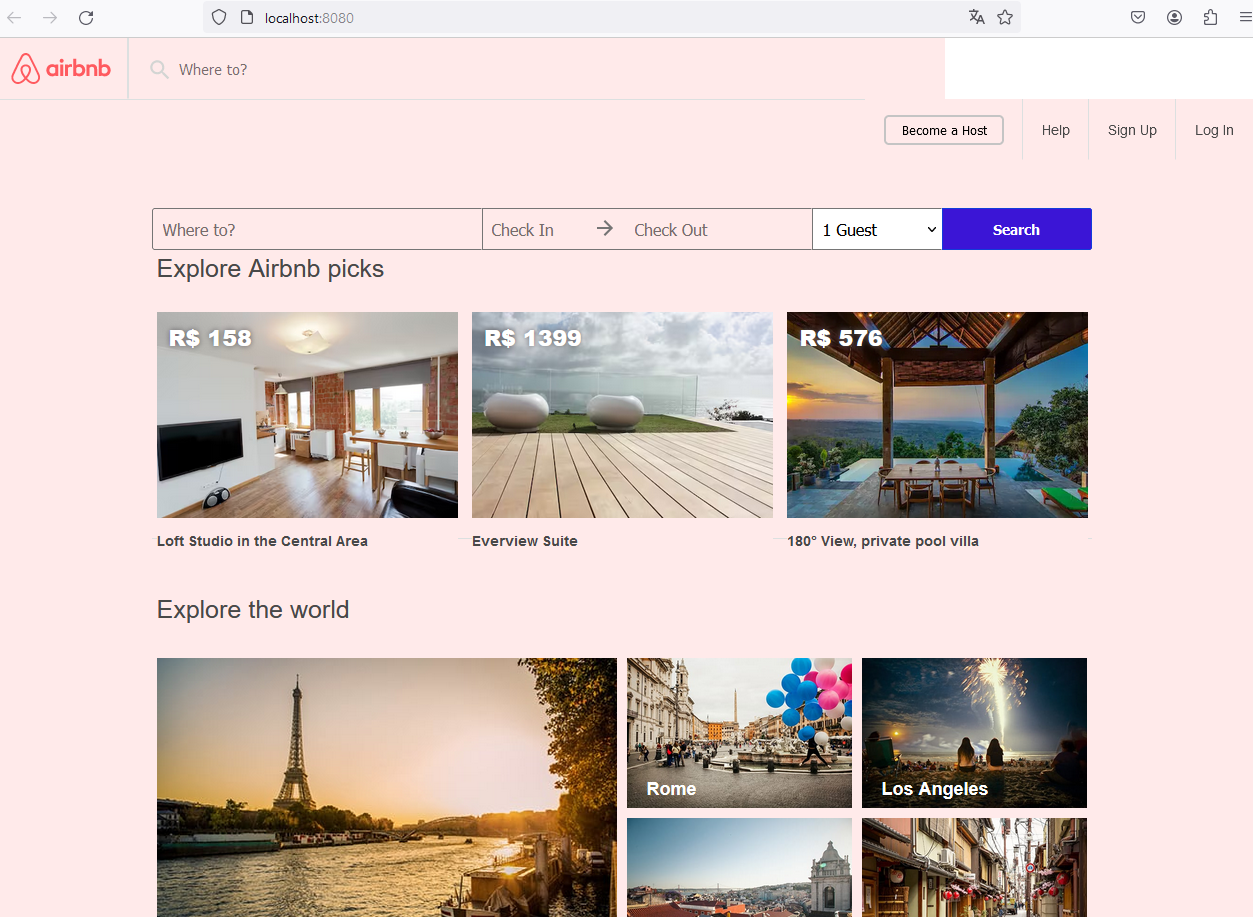
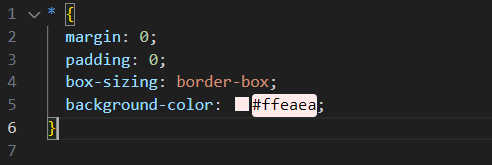
**(web5test) pero en esta ocasión enlaza o mapea la ruta del host que contiene los estáticos de la web con el directorio del contenedor desde el que se sirven Este paso lo puedes realizar directamente por consola utilizando el comando docker run con la imagen oficial de httpd (es decir, puedes prescindir de la imagen custom construida a partir del Dockerfile). Comprueba que puedes acceder al sitio web que acabas de servir en localhost**



1. **Sin eliminar o detener el contenedor anterior, modifica el fichero html y**

**verifica que el cambio se aplica en el sitio web (puede ser necesario que actualices la página) ¿Se aplica el cambio?¿Por qué?**



* 1. Ocurre que al colocar un volumen. Los cambios realizados en los archivos del host se reflejan dentro del contenedor y persisten incluso después de que el contenedor se detenga o se elimine y se conserva a lo largo del ciclo de vida del contenedor.