**Docker storage** refers to the management and persistence of data within containers. Containers are stateless and ephemeral, which means any data created within them can be lost when the container stops. Docker provides various storage options to ensure data persistence and manage application states effectively.

Two type of storage:

1. **Non-persistent**: data resides with the container , gets deleted when container deleted, By default all container has it.

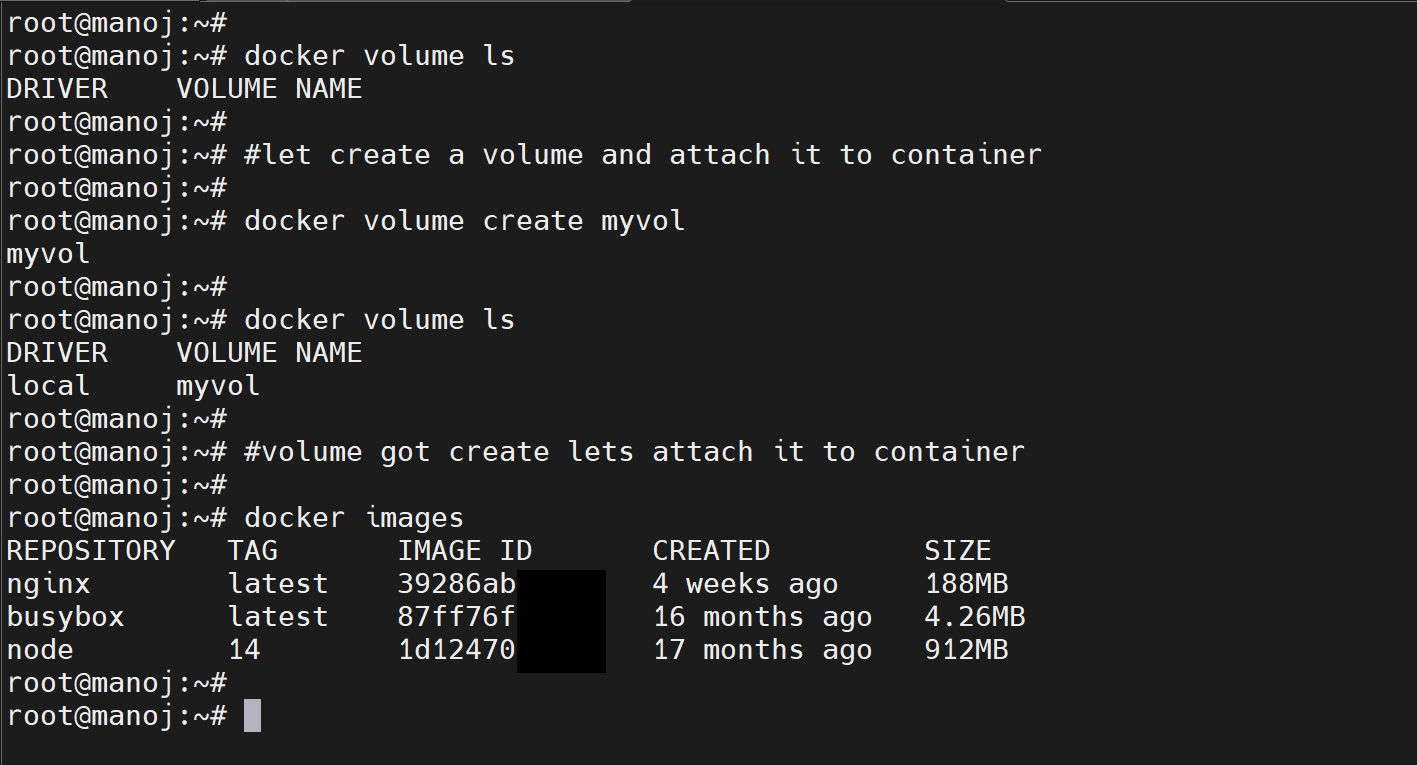
* Storage Drivers: Overlay2 (Default for Most Linux Systems)

1. **Persistent**: data doesn’t resides within the container, doesn’t get deleted when container deleted.

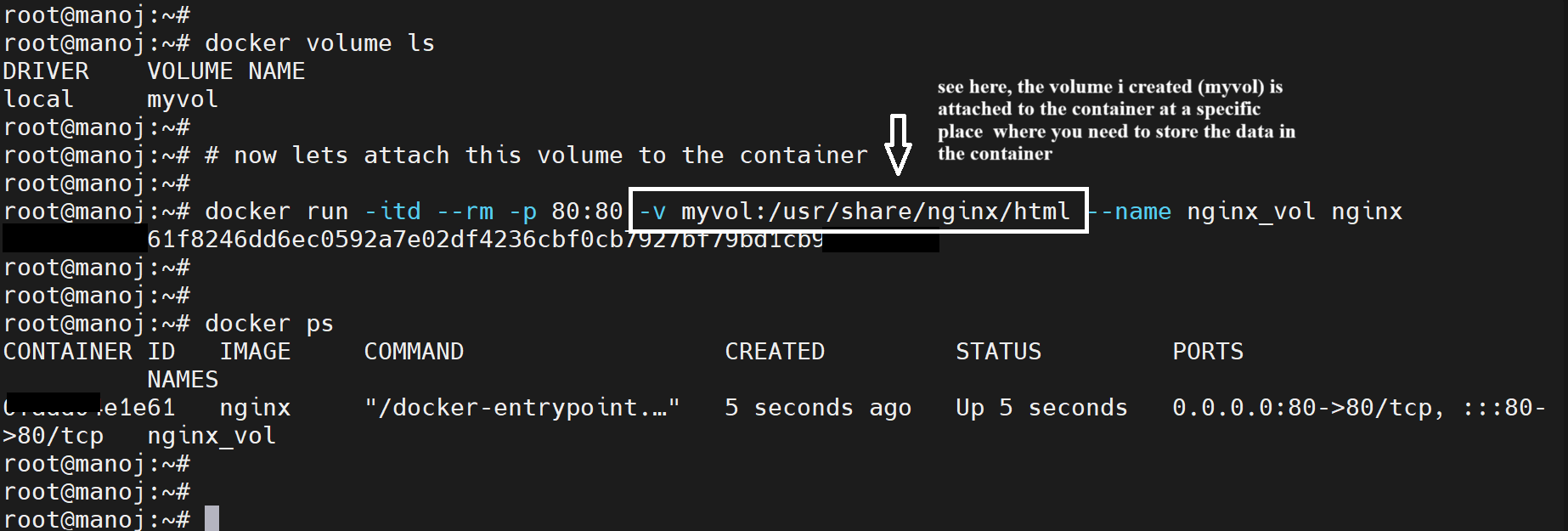
* **Types of persistent storage**: Volume and Bind mount

**Volumes**: Managed by docker and stored outside of the container filesystem. Volume are the best way to persist the data. Volumes can be shared between containers, backed up, restored, and mounted as read-only or read-write.

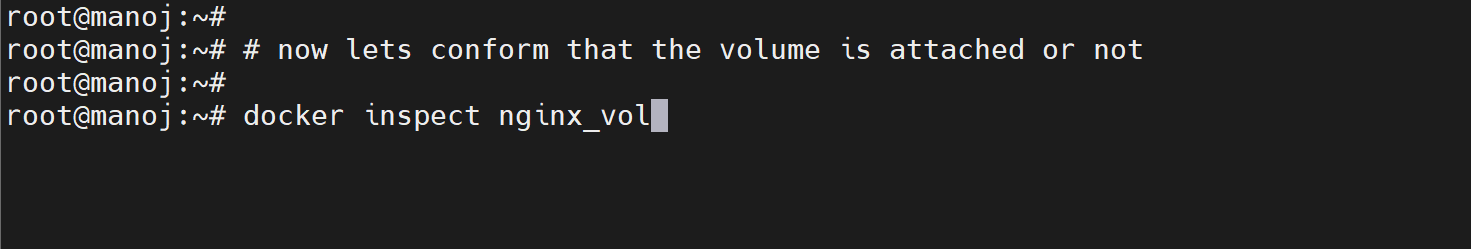
**Creation of docker volume**

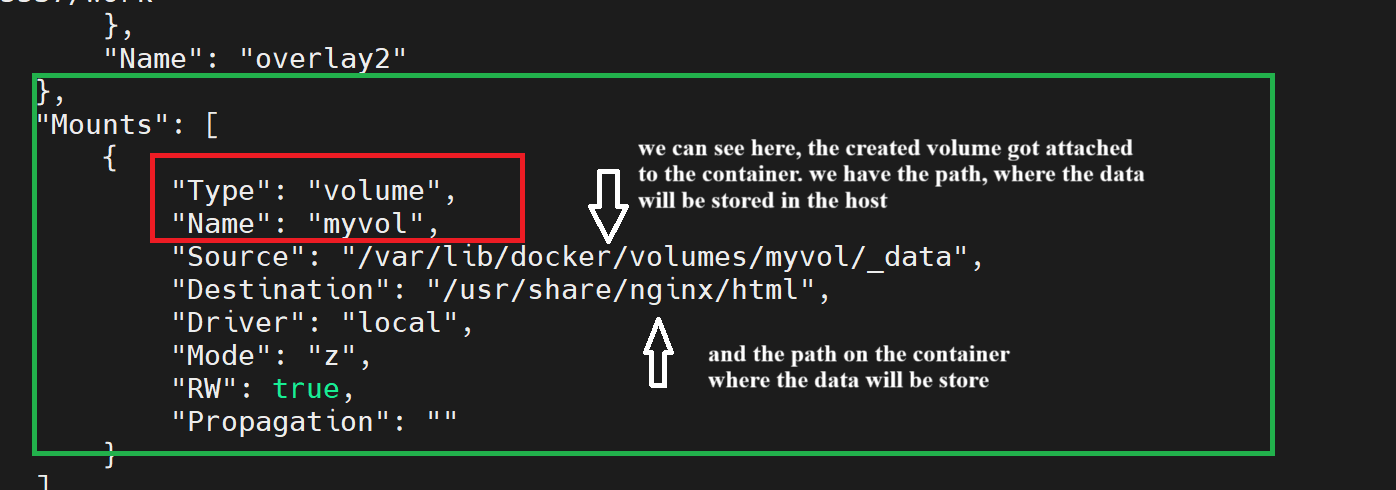


Mounting the created volume on to the container and providing the path where I want to store the volume in the container.

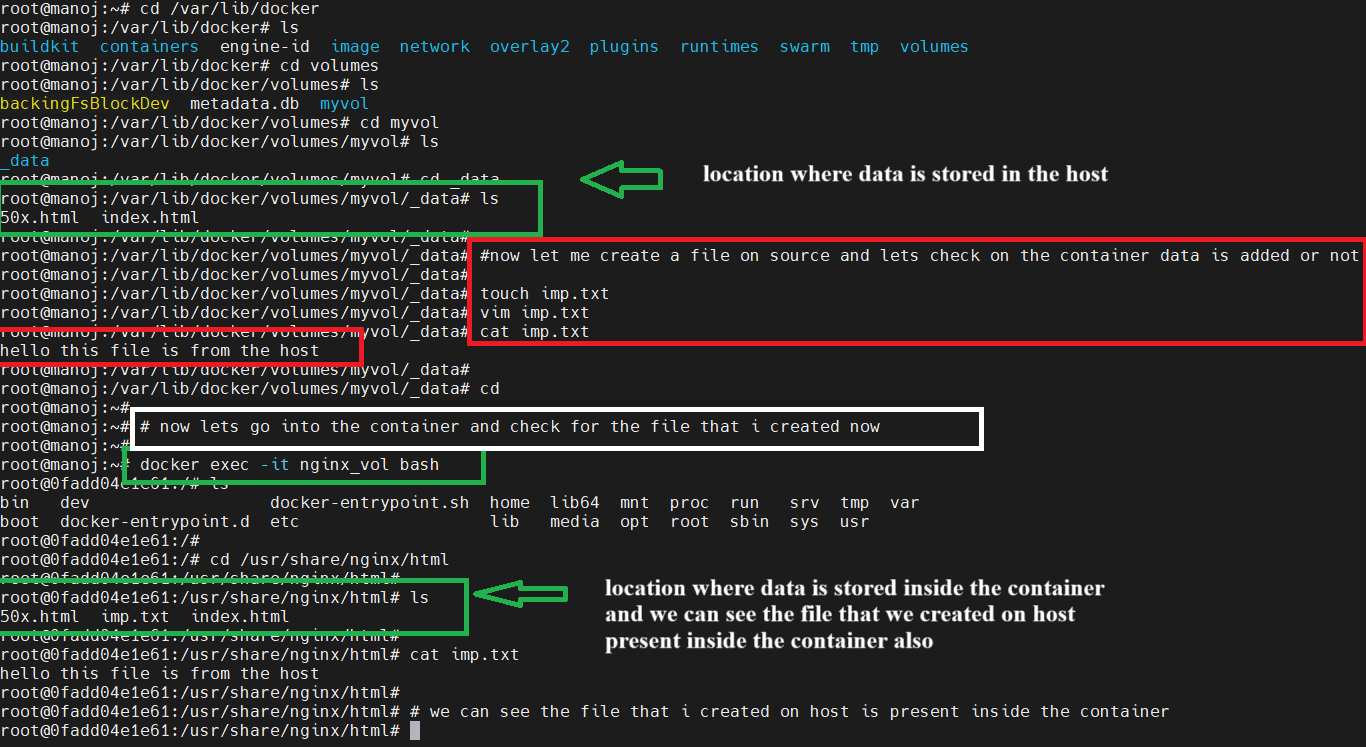


Using Docker inspect we can check volume type that mounted onto the container

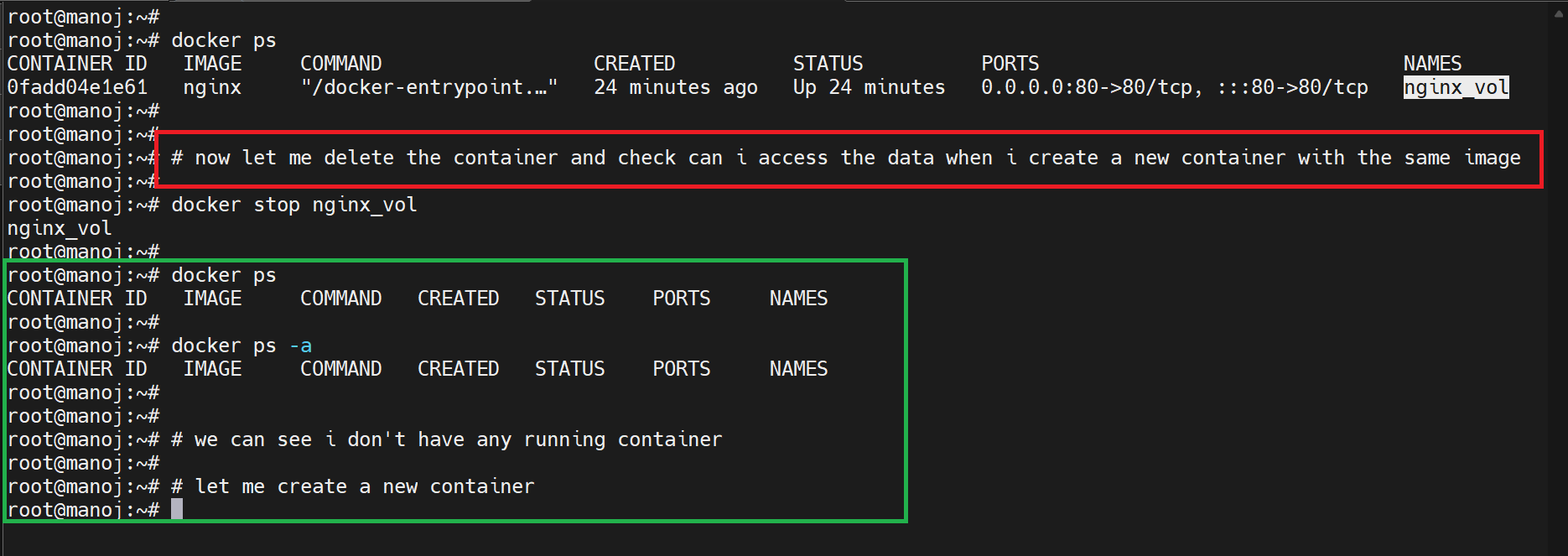




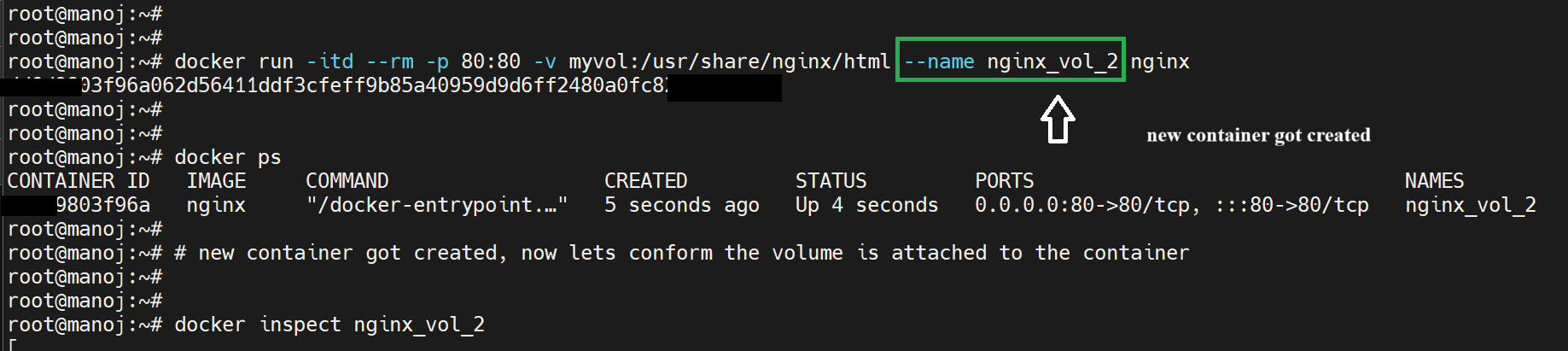
Location where data will be stored In the Host: /**var/lib/docker/volumes/ and** I created a file in host location and we can see that same data is present in the container location also



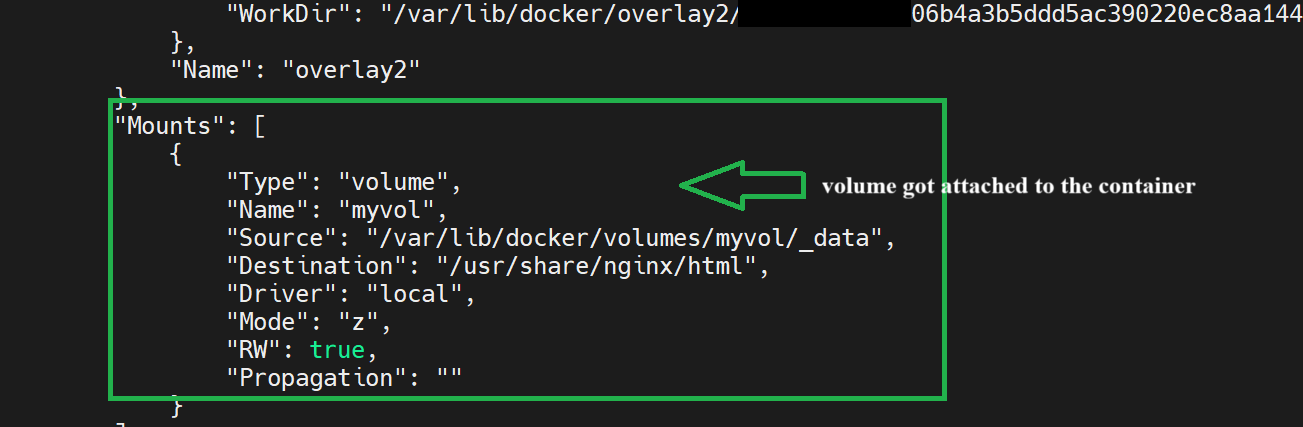
I will delete the docker container and see can we still get our data



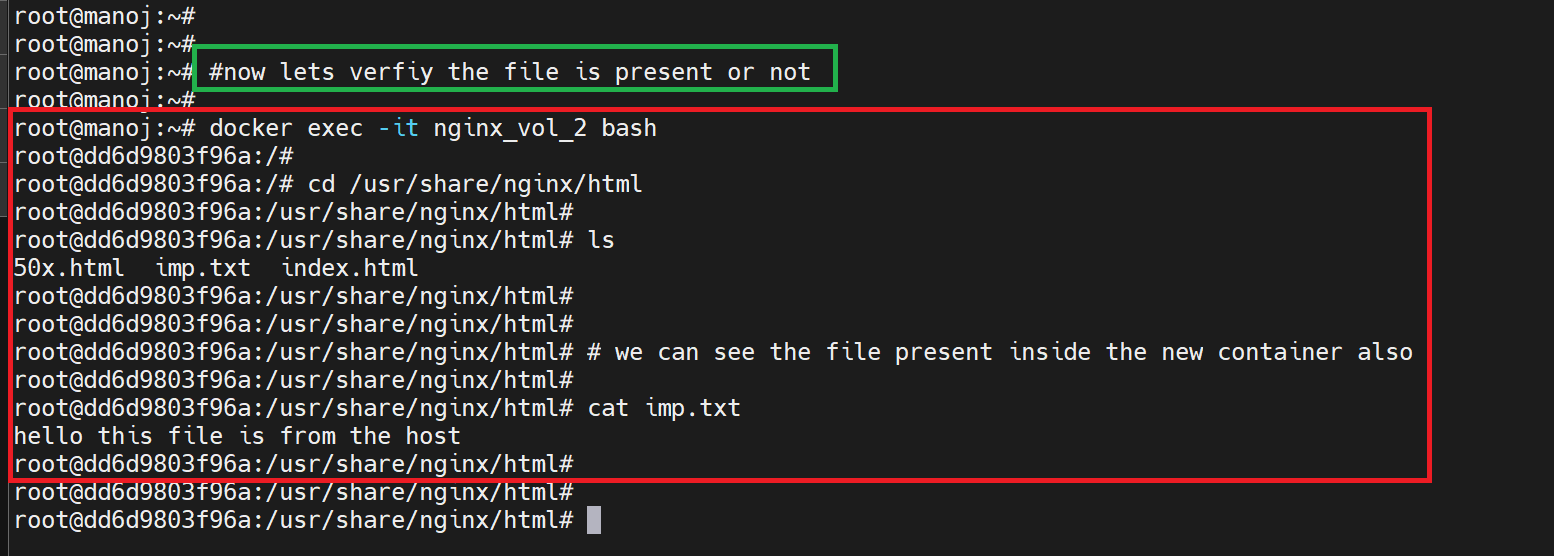
Creating the new container and mounting the volume that we created earlier.



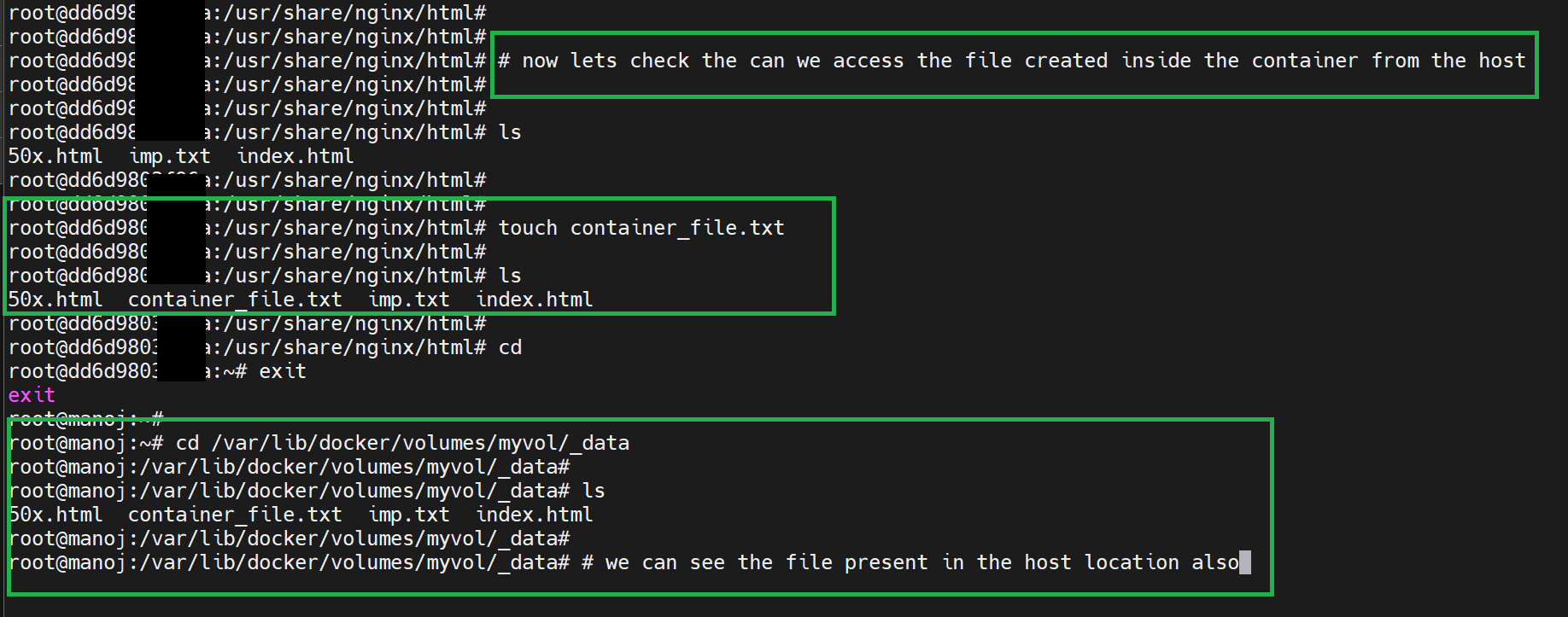
We can verify using docker inspect that volume got attached to the container.



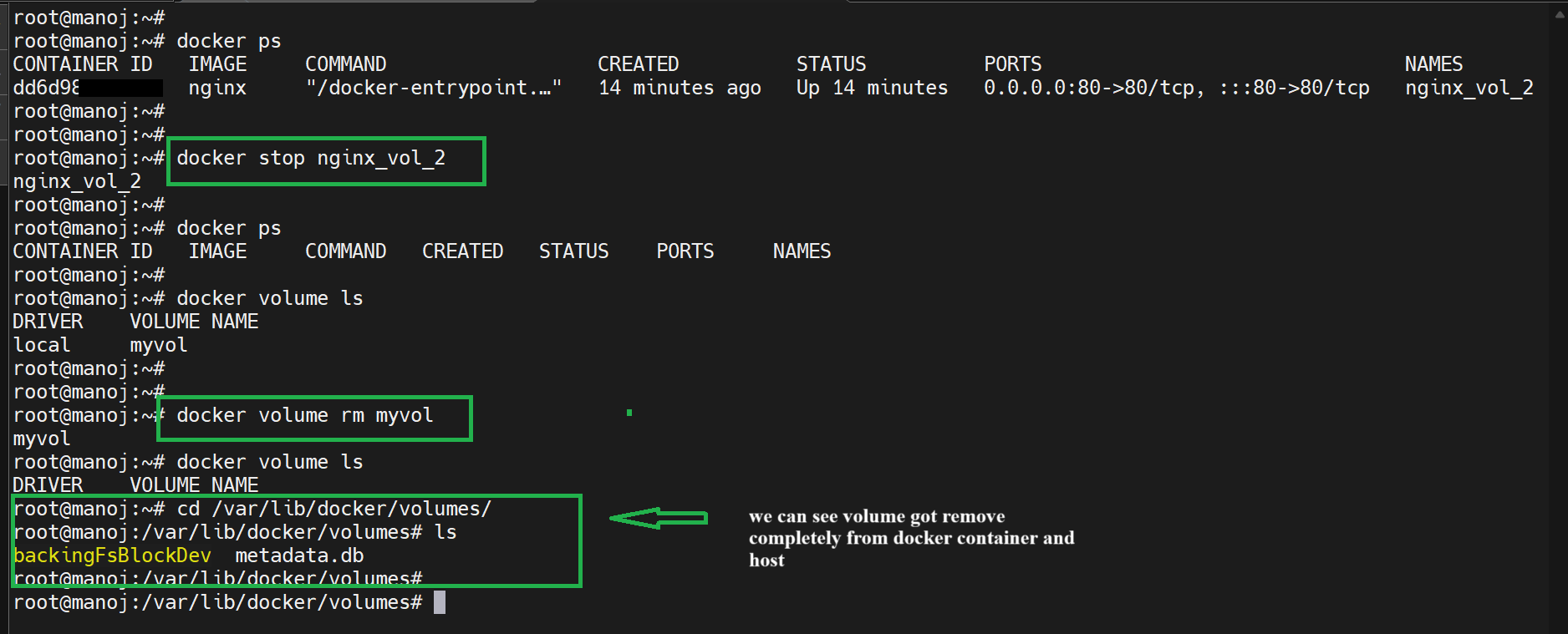
Now lets verify the files are still present or not. we can see that files are still present in the host, even after deleting the container.



Docker container also holds the files inside the container.

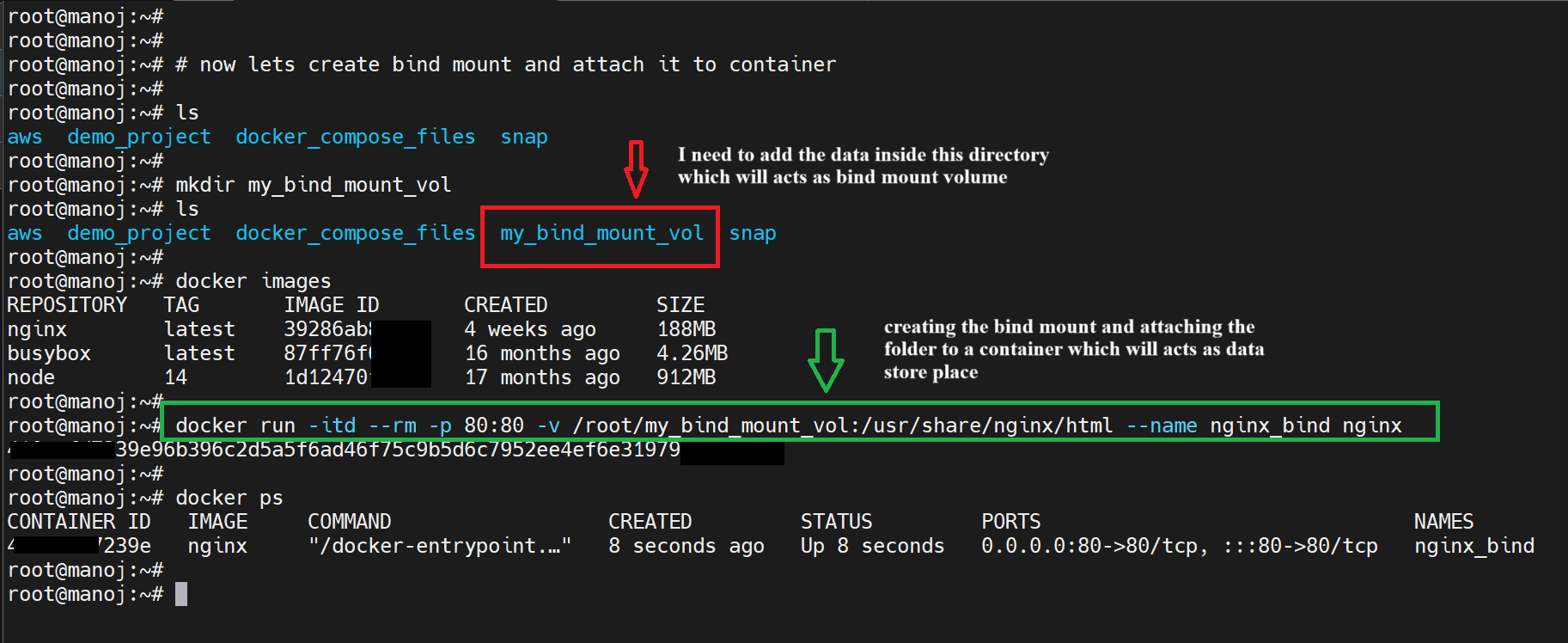


Remove the docker volume



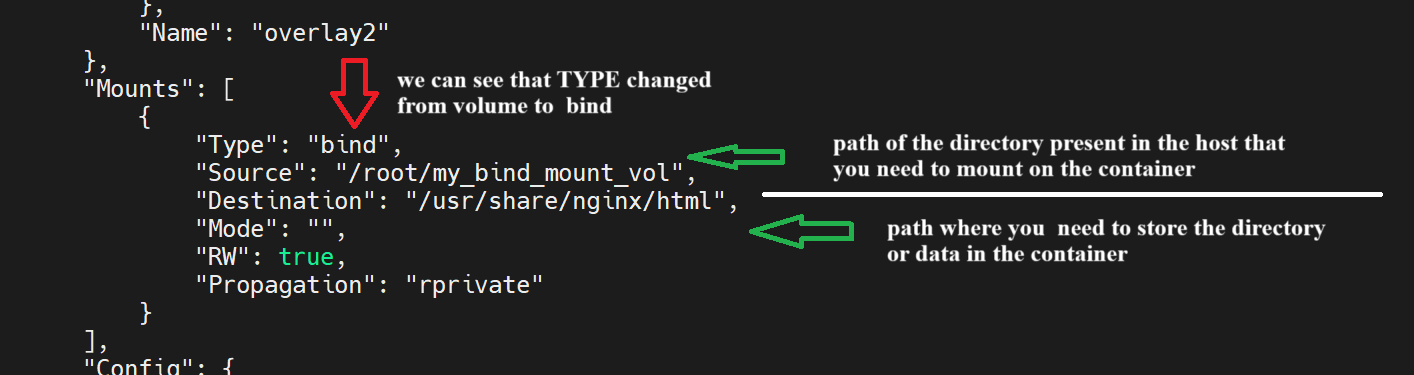
**Bind Mount:** Bind mounts allow containers to access specific directories on the host’s filesystem, outside the Docker-managed area. Ideal for development environments where you need real-time access to host files.

Creating the director to mount on the container.

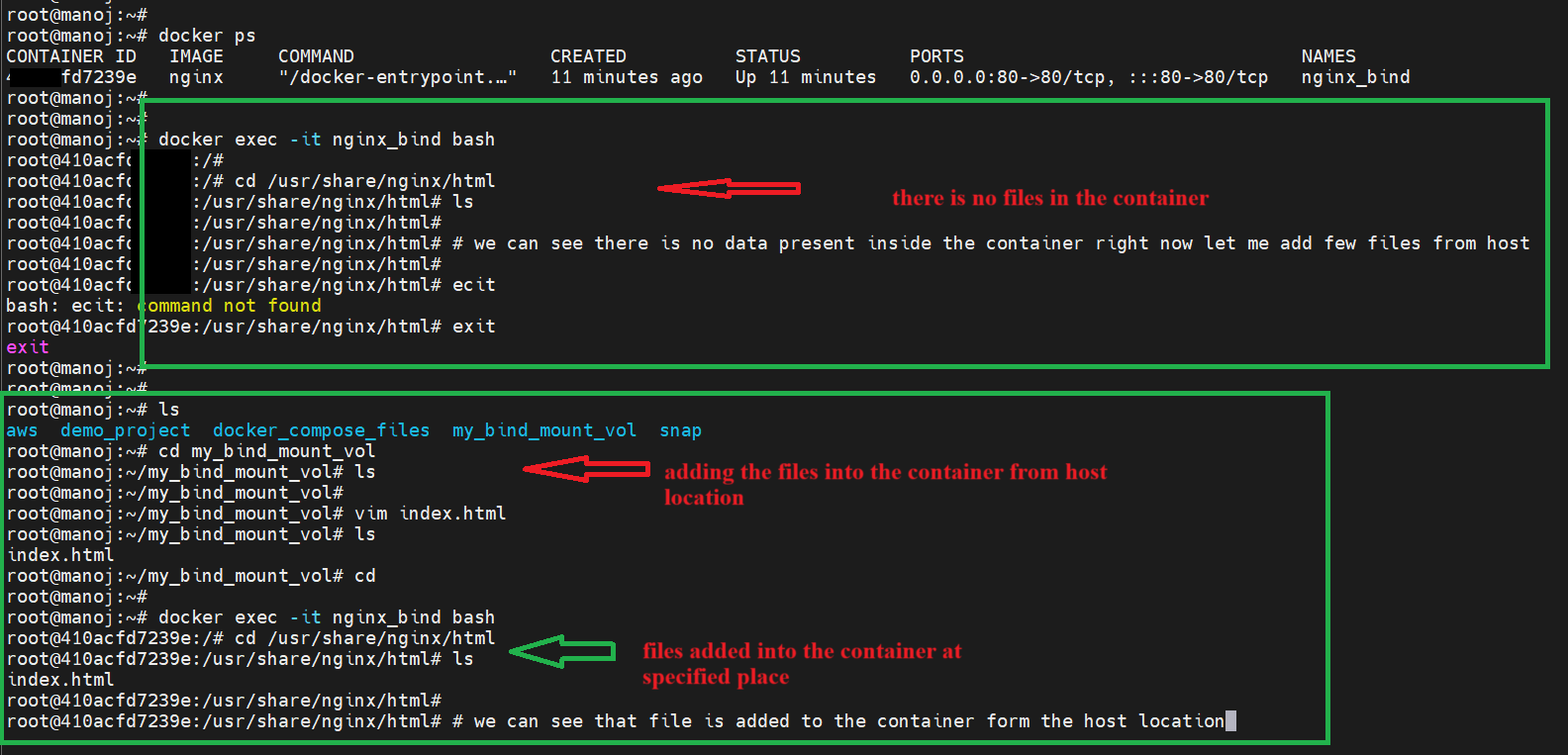


Now we can see here type of mount changes to bind, this show that we mounted bind volume onto the docker container.

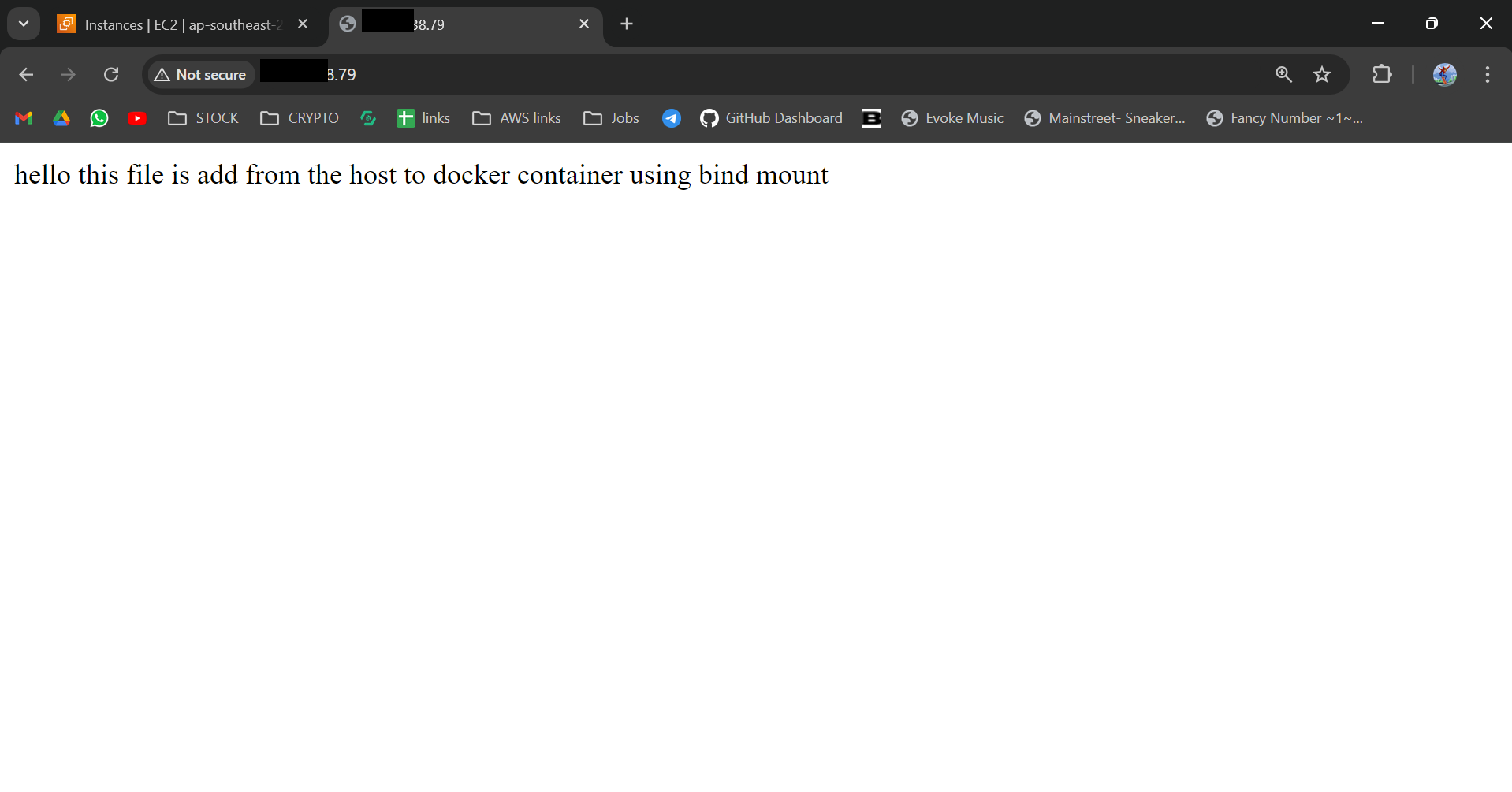
Even we can see the source location present on the host and Destination location present on the container here.



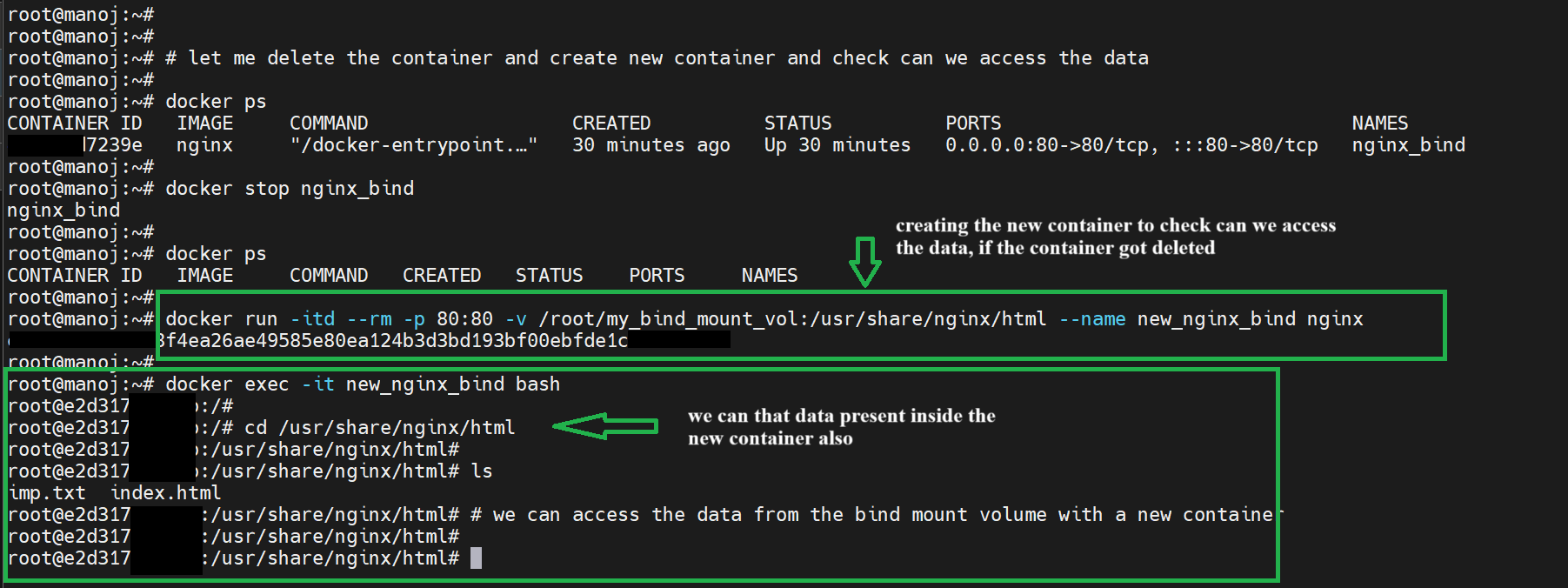
Now let me add file in the host location and check can we see same file in docker bind mount location also.



We can see that, I an access the data from the bind mount location



Deleted the current container and built new container to check still can we access the data present in bind mount directory.



And we can see still I can access the data, if a bind mount directory on the host is deleted, the container will lose access to the data in that directory. The container will still run, but attempts to read or write to the bind-mounted path will fail or result in an empty directory.

