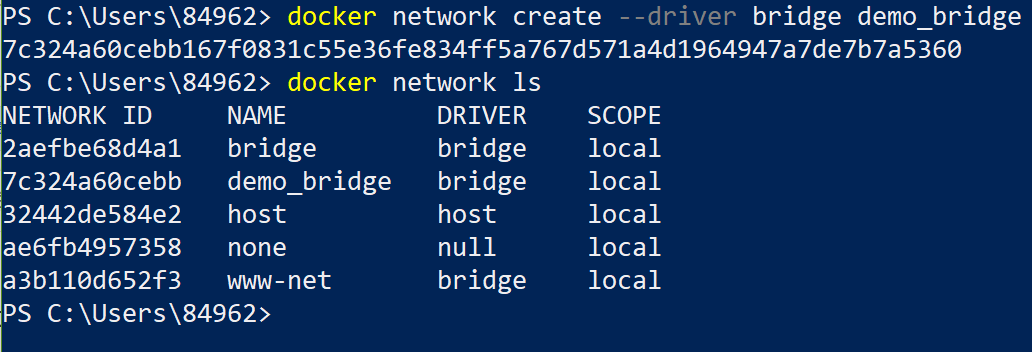
# **Demo thực hành network & storage trong Docker**

# **Demo network Docker**

## **1: Demo mạng bridge**

* Input:
* 1 môt network bridge riêng
* 2 container chạy ubuntu
* Output:
* 2 container ping được cho nhau và ping đc ra mạng ngoài

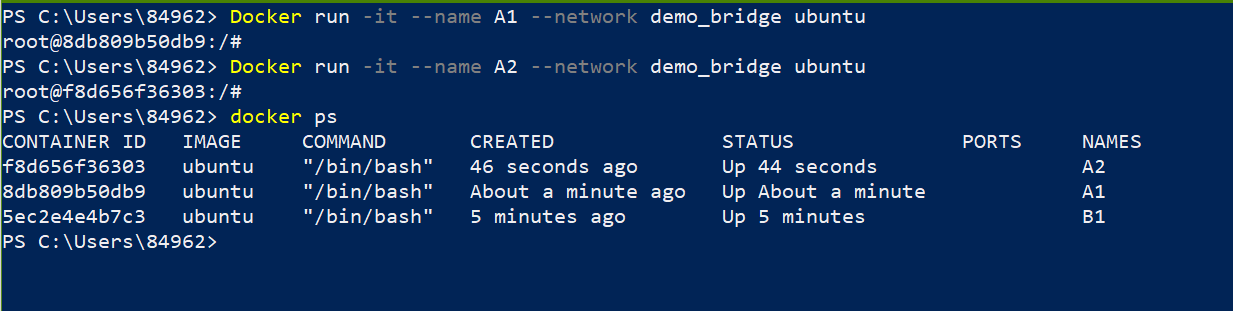
**Step 1:** Tạo một network bridge



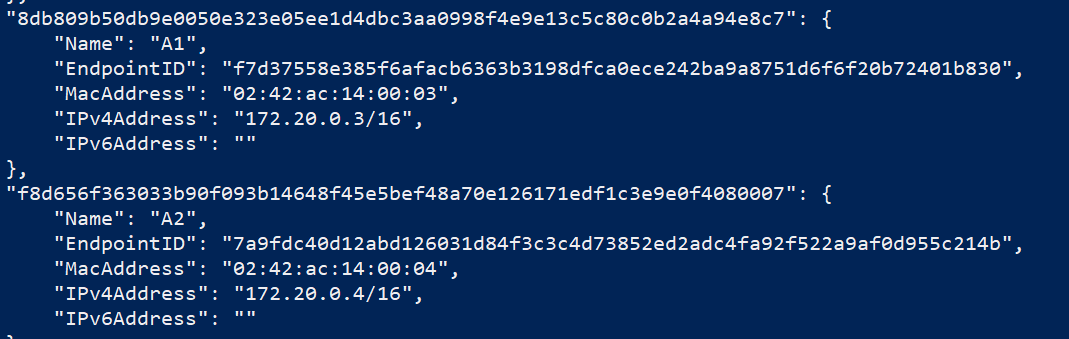
* Sử dụng câu lệnh:docker network create --driver bridge demo\_bridge
* Kiểm tra xem những network đã có: docker network ls

**Step 2:** Tạo 2 container cùng kết nối với mạng bridge vừa tạo

* Tạo 2 container A1, A2 bằng Image ubuntu, TAG: latest

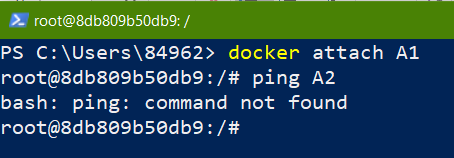


* Sử dụng câu lệnh:docker run –it –name A1 –network demo\_bridge ubuntu
* Sử dụng câu lệnh:docker run –it –name A2 –network demo\_bridge ubuntu
* Kiểm tra xem 2 container đã cùng 1 mạng chưa và xem đia chỉ ip các container: docker network inspect demo\_bridge

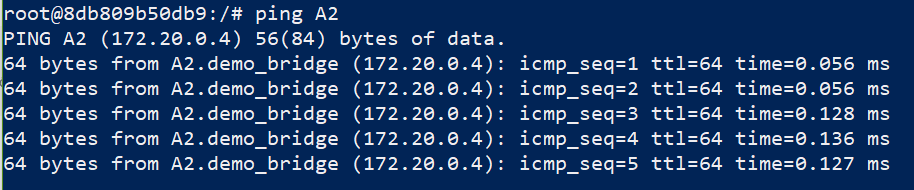


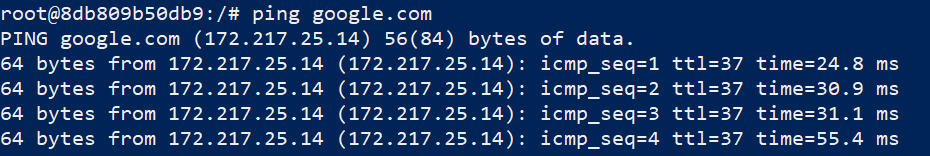
**Step 3:** Ping từ A1 sang A2 và Ping địa chỉ google.com

* Truy câp vào A1: docker attach A1
* Ping từ A1 sang A2



* Do chỉ mới ubuntu thì chưa có Linux ping command ta sẽ vào A1 cài Linux ping command với câu lệnh: apt-get update && apt-get install -y iputils-ping Khi đã cài xong ta tiếp tục Ping sang A2 và Ping tới địa chỉ google.com



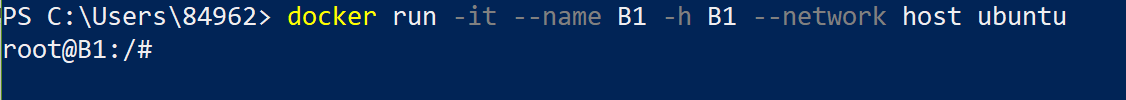


* Như vậy chúng ta đã hoàn thành demo mạng bridge, kết quả là chúng ta có thể sử dụng mạng Bridge để gọi tới các container trong cùng dải mạng và có thể gọi tới các địa chỉ bên ngoài internet

## **2: Demo mạng host**

* Input:
* 1 môt network host
* 1 container chạy ubuntu có cài Linux ping command
* Output:
* container ping đc ra mạng internet

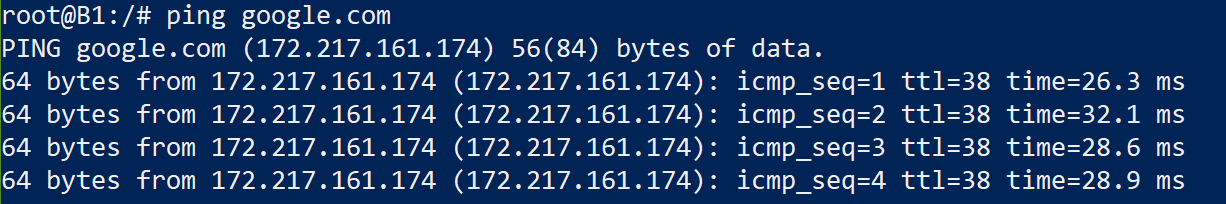
step 1: Tạo một contrainer B1 và sửa dụng network host

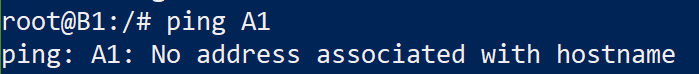


* Cài thêm Linux ping command cho container B1

**Step 2:** Ping A1, và Ping tới google.com

* Sau đó ping từ B1 vào địa chỉ google.com vaf Ping sang A1



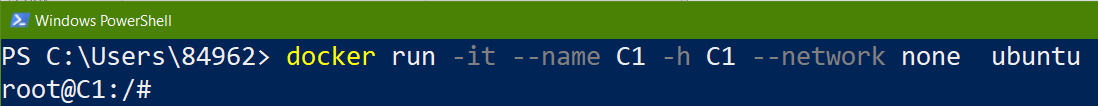


* Như vậy với mạng host thì ta có thể kết nối đc với mạng internet nhưng ko thể kết nối đc với dải mạng của container khác trong máy host

## **3: Tạo một mạng none**

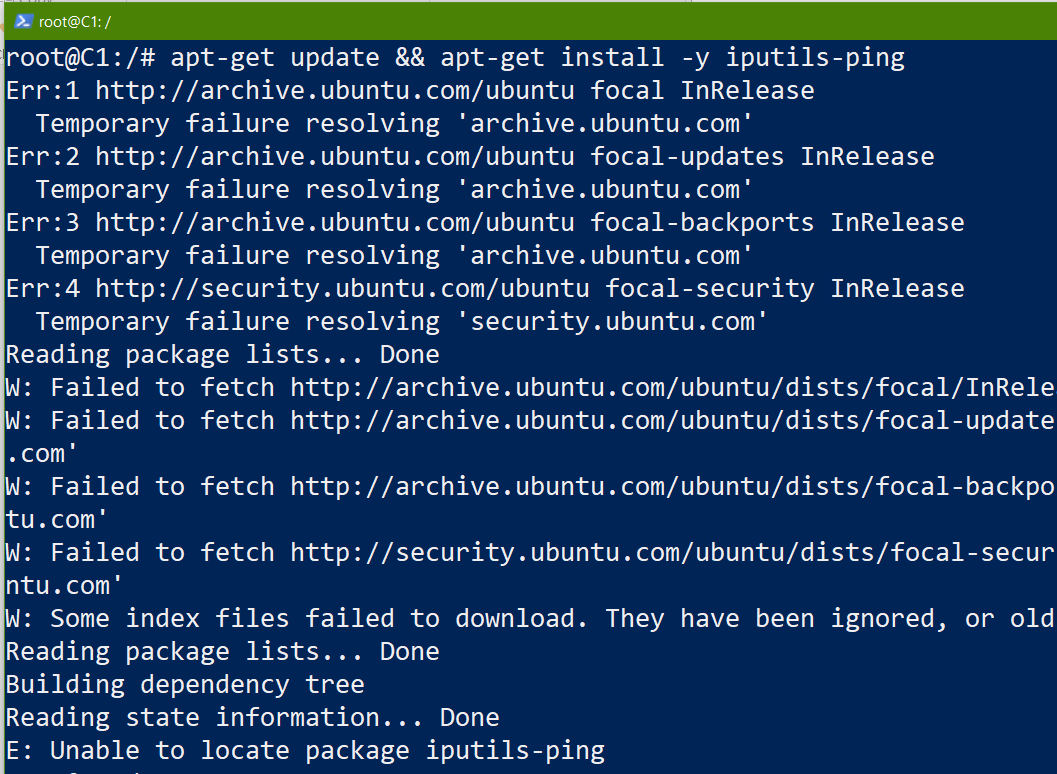
* Input:
* 1 môt network none
* 1 container chạy ubuntu
* Output:
* container không ping đc ra mạng internet và các container khác

**Step 1:** tạo một Container dùng mạng none



**Step 2:** Cài Linux ping command cho C1

* Do không thể kết nối được với mạng internet lên nó hiện lỗi



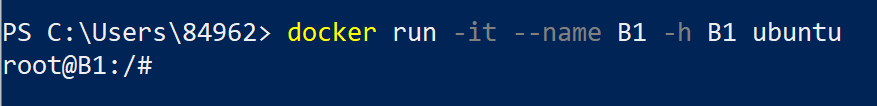
* Như vậy với mạng none ta chỉ có thể tạo ra được các container cô lập, không có liên kết với mạng ngoài và các container khác

# **Demo Storage Docker**

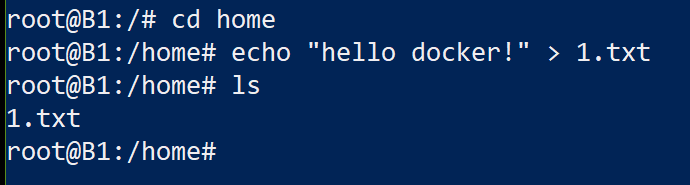
## **1: Demo Storage Stateless**

* Input:
* 1 container chạy ubuntu
* Output:
* File data trong container không được lưu trữ

**Step 1:** Tạo một container B1 từ Image ubuntu



* Sau đó tạo file 1.txt bằng lệnh: echo "hello docker" > 1.txt trong thư mục home trên ubuntu
* Kiểm tra file đã tạo: ls



* Sau đó xóa container đi thì file 1.txt trong đó cũng bị mất
* Như vậy với Stateless thì data sẽ bị mất đi khi container bị xóa

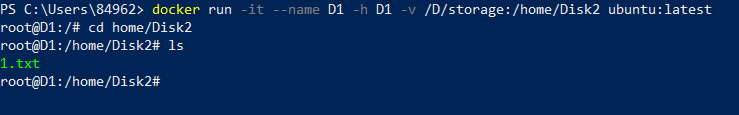
## **2: Demo Storage Stateful**

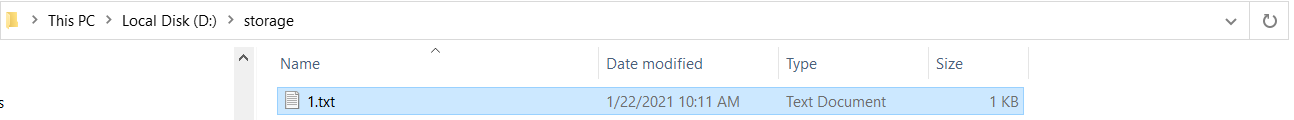
### **2.1: Demo bind mounts**

* Input:
* 1 ổ đĩa trên Docker có ánh xạ tới một folder trong máy host
* 2 container chạy ubuntu, cùng sử dụng ổ đĩa
* Output:
* Xóa Container đi thì data vẫn con và có thể đọc từ Container khác

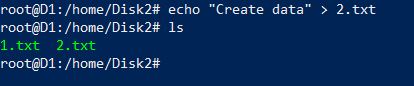
**Step 1:** Tạo một Contrainer có mount tới 1 folder trên máy host(/D/storage)

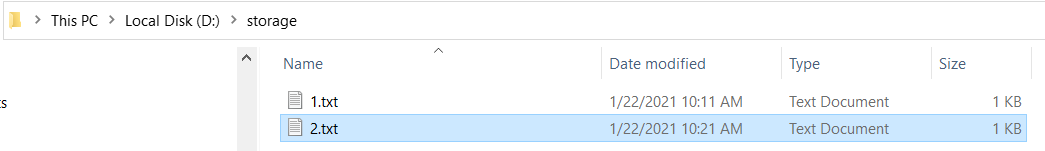
* Tạo Contrainer mount tới 1 folder trên máy host: docker run -it --name D1 -h D1 -v /D/storage:/home/Disk2 ubuntu:latest
* Và truy cập vào folder Disk2 để xem các file trong đó. Fife 1.txt được tạo từ máy host



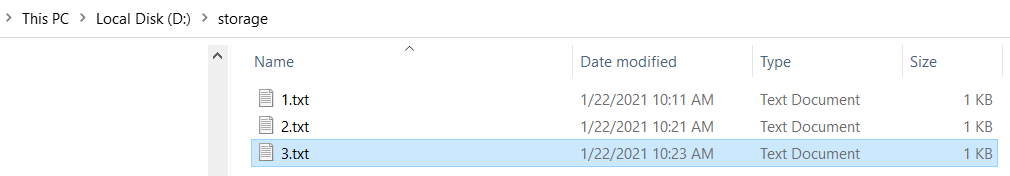


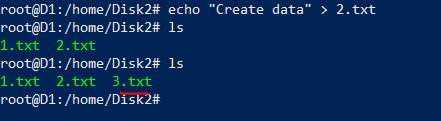
* Sau đó tạo một file 2.txt trong Container



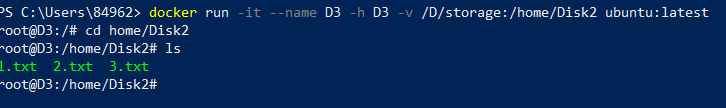


* Tạo thử file 3.txt từ máy host, kiểm tra folder Disk2 của Container





**Step 2:** Xóa Contrainer vừa tạo và tạo một Container mới có mount tới folder /D/storage, kiểm tra lại data



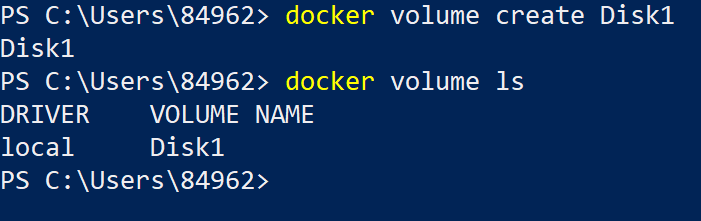
* Như vậy với kiểu lưu trữ này thì cũng ta có thể lưu được các data được tạo từ máy host cũng như từ Container vào một folder đã được chỉ định

### **2.2: Demo volume**

* Input:
* 1 ổ đĩa trên Docker
* 2 container chạy ubuntu, cùng sử dụng ổ đĩa
* Output:
* Xóa Container đi thì data vẫn con và có thể đọc từ Container khác

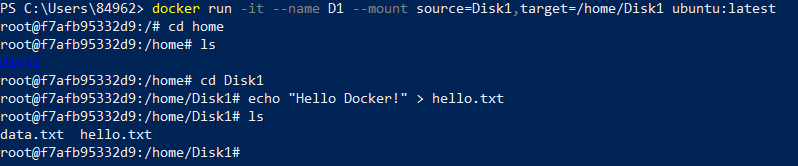
**Step 1:** Tạo một ổ đĩa trên Docker, kiểm tra lại ổ đĩa đã tạo

* Tạo một ổ đĩa: docker volume create Disk1
* Kiểm ta ổ đĩa vừa tạo: docker volume ls

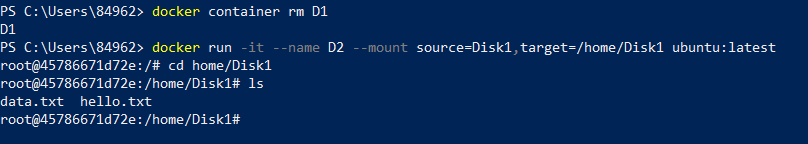


**Step 2:** Tạo một container rồi mount một folder tới volume DISK1

* Tạo một container sử dụng ổ đĩa vừa tạo: docker run -it --name D1 --mount source=Disk1,target=/home/Disk1 ubuntu:latest



**Step 3:** Xóa container vừa tạo, tạo một container mới mount tới DISK1. Kiểm tra file đã tạo từ container bị xóa



* Như vậy với cách lưu trữ này thì dù Container bị xóa thì data cũng không mất đi, và có thể lấy, chia sẻ data từ container khác. Volume dược quản lý bởi docker, sẽ đảm bảo an toán hơn bind mount. Khi mà bind mount phải xác định một folder trên máy host, sẽ gây ra nhiều nguy cơ về an toàn thông tin.