**1. รายชื่อคอนเทนเนอร์ที่กำลังทำงานอยู่**

ก่อนอื่น ให้ตรวจสอบว่าคอนเทนเนอร์ใดกำลังทำงานอยู่:

Bash

docker ps

* คำสั่งนี้จะแสดงคอนเทนเนอร์ที่กำลังทำงาน เช่น:

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

a1b2c3d4e5f6 mcr.microsoft.com/mssql/server:2019-latest "/opt/mssql/bin/sqlservr" 2 hours ago Up 2 hours 0.0.0.0:1433->1433/tcp mssql

7g8h9i0j1k2l node:18 "node server.js" 2 hours ago Up 2 hours 0.0.0.0:6000->5000/tcp backend

**2. เข้าไปในคอนเทนเนอร์**

ใช้คำสั่ง docker exec เพื่อเข้าไปในคอนเทนเนอร์ที่ต้องการ:

Bash

docker exec -it <container\_name> /bin/bash

ตัวอย่าง:

bash

docker exec -it mssql /bin/bash

* หากคอนเทนเนอร์ไม่มี /bin/bash (เช่น บางคอนเทนเนอร์อาจใช้ sh):

bash

docker exec -it <container\_name> /bin/sh

* เมื่อเข้าสู่คอนเทนเนอร์แล้ว คุณสามารถใช้คำสั่ง Linux เพื่อดูหรือจัดการระบบ เช่น:

bash

ls

pwd

cat <file\_name>

**3. ออกจากคอนเทนเนอร์**

เมื่อต้องการออกจากคอนเทนเนอร์ ให้ใช้คำสั่ง:

bash

exit

**4. ดู Log ของคอนเทนเนอร์**

หากต้องการดูบันทึกหรือ log ของคอนเทนเนอร์:

bash

docker logs <container\_name>

ตัวอย่าง:

bash

docker logs backend

* ใช้ -f เพื่อดู log แบบเรียลไทม์:

bash

docker logs -f backend

**5. รันคำสั่งภายในคอนเทนเนอร์โดยไม่เข้าไป**

หากคุณต้องการรันคำสั่งในคอนเทนเนอร์โดยไม่เข้าไป คุณสามารถใช้:

bash

docker exec -it <container\_name> <command>

ตัวอย่าง:

bash

docker exec -it backend node server.js

**6. ดูรายละเอียดของคอนเทนเนอร์**

หากต้องการดูข้อมูลเพิ่มเติมเกี่ยวกับคอนเทนเนอร์ (เช่น พอร์ต, Volume ที่ติดตั้ง):

bash

docker inspect <container\_name>

ในการคัดลอกไฟล์ใน Docker คุณสามารถใช้คำสั่ง docker cp โดยมีรูปแบบดังนี้:

**การคัดลอกจากโฮสต์ไปยังคอนเทนเนอร์**

bash

docker cp <path-to-local-file> <container-id>:<path-to-destination>

ตัวอย่าง:

bash

docker cp /path/to/local/file.txt my\_container:/path/in/container/file.txt  
  
docker cp /Users/sommanutketpong/Downloads/testreact/tong.txt testreact-frontend-1:/app/src/tong2.txt

**การคัดลอกจากคอนเทนเนอร์ไปยังโฮสต์**

bash

docker cp <container-id>:<path-to-file-in-container> <path-to-local-destination>

ตัวอย่าง:

bash

docker cp my\_container:/path/in/container/file.txt /path/to/local/file.txt

docker cp txt testreact-frontend-1:/app/src/tong2.txt /Users/sommanutketpong/Downloads/testreact/tong.

เพียงแค่ระบุเส้นทางที่คุณต้องการคัดลอกไฟล์จากหรือไปยังคอนเทนเนอร์หรือโฮสต์

การคัดลอกไฟล์ระหว่างคอนเทนเนอร์ Docker (เช่น จาก docker1 ไปยัง docker2) ไม่สามารถทำได้โดยตรงผ่านคำสั่ง docker cp เพราะ docker cp ใช้สำหรับคัดลอกไฟล์จากคอนเทนเนอร์ไปยังโฮสต์ หรือจากโฮสต์ไปยังคอนเทนเนอร์เท่านั้น แต่คุณสามารถทำตามขั้นตอนนี้เพื่อคัดลอกไฟล์ระหว่างคอนเทนเนอร์:

**วิธีที่ 1: ใช้คอนเทนเนอร์หนึ่งเป็นตัวกลาง**

1. **คัดลอกไฟล์จาก docker1 ไปยังโฮสต์:**

bash

docker cp docker1:/path/to/file /path/to/local/directory

1. **คัดลอกไฟล์จากโฮสต์ไปยัง docker2:**

bash

docker cp /path/to/local/directory/file docker2:/path/to/destination

**วิธีที่ 2: ใช้ Docker Volume หรือ Network Share**

หากไฟล์อยู่ในพื้นที่ที่ใช้ร่วมกัน (shared volume) หรือเชื่อมต่อกับเครือข่าย (network), คุณสามารถให้ทั้งสองคอนเทนเนอร์เข้าถึงไฟล์ใน volume หรือแชร์โฟลเดอร์เดียวกันได้

1. สร้าง volume หรือ mount directory ที่ใช้ร่วมกัน
2. ให้ทั้งสองคอนเทนเนอร์เข้าถึง volume หรือ directory เดียวกันเพื่อถ่ายโอนไฟล์

**วิธีที่ 3: ใช้คำสั่ง docker exec เพื่อคัดลอกไฟล์ระหว่างคอนเทนเนอร์**

1. ใช้คำสั่ง docker exec เพื่อคัดลอกไฟล์จาก docker1 ไปยัง docker2 ผ่านการสั่งให้ docker1 ส่งไฟล์ไปยัง docker2โดยตรง:

bash

docker exec docker1 cat /path/to/file | docker exec -i docker2 tee /path/to/destination > /dev/null

docker exec testreact-frontend-1 cat /app/tong.txt | docker exec -i testreact-backend-1 tee/app/src/tong3.txt > /dev/null

วิธีนี้จะทำให้คุณสามารถคัดลอกไฟล์จากคอนเทนเนอร์หนึ่งไปยังอีกคอนเทนเนอร์หนึ่งโดยตรงได้.

ในการเข้าถึงคอนเทนเนอร์ Docker จากภายนอก (จากโฮสต์หรือเครื่องอื่น) คุณสามารถใช้หลายวิธี เช่น การเข้าไปในคอนเทนเนอร์โดยตรงด้วย docker exec หรือการเปิดพอร์ตเพื่อเข้าถึงบริการที่รันอยู่ในคอนเทนเนอร์ผ่านเครือข่าย

**วิธีที่ 1: เข้าถึงคอนเทนเนอร์ผ่าน docker exec**

1. **เข้าสู่คอนเทนเนอร์ที่กำลังรันอยู่:**

bash

docker exec -it <container\_id\_or\_name> /bin/bash

หรือถ้าคอนเทนเนอร์ใช้ sh แทน bash:

bash

docker exec -it <container\_id\_or\_name> /bin/sh

1. หากต้องการใช้คำสั่งภายในคอนเทนเนอร์ คุณสามารถพิมพ์คำสั่งในเซสชั่นที่เปิดขึ้นมา:

bash

docker exec -it <container\_id\_or\_name> <command>

**วิธีที่ 2: เข้าถึงคอนเทนเนอร์ผ่านพอร์ตที่เปิดไว้ (สำหรับบริการที่รันในคอนเทนเนอร์)**

1. **เปิดพอร์ตในคอนเทนเนอร์:** เมื่อตอนรันคอนเทนเนอร์ด้วย docker run, คุณสามารถเปิดพอร์ตให้สามารถเข้าถึงบริการจากภายนอกได้ เช่น:

bash

docker run -p 8080:80 <image\_name>

คำสั่งนี้จะเปิดพอร์ต 8080 บนโฮสต์และแมปไปยังพอร์ต 80 ของคอนเทนเนอร์

1. **เข้าถึงบริการจากภายนอก:** หลังจากเปิดพอร์ต, คุณสามารถเข้าถึงบริการที่รันในคอนเทนเนอร์จากเบราว์เซอร์หรือเครื่องอื่น ๆ ที่เชื่อมต่อกับโฮสต์ผ่าน URL เช่น:

arduino

http://<host\_ip>:8080

**วิธีที่ 3: เข้าถึงคอนเทนเนอร์ผ่าน Docker Network**

หากคอนเทนเนอร์ของคุณอยู่ใน Docker network เฉพาะ, คุณสามารถเชื่อมต่อไปยังคอนเทนเนอร์อื่นๆ ผ่าน Docker network ได้ โดยใช้ชื่อคอนเทนเนอร์เป็น DNS (เช่น docker2 สามารถใช้ได้ในคอนเทนเนอร์ docker1 ถ้าทั้งสองอยู่ใน network เดียวกัน).

bash

docker network connect <network\_name> <container\_id\_or\_name>

การเข้าถึงคอนเทนเนอร์จากภายนอกขึ้นอยู่กับว่าคุณต้องการเข้าถึงคอนเทนเนอร์ในลักษณะไหน เช่น การเข้าถึงการใช้งานภายในคอนเทนเนอร์หรือการเข้าถึงบริการที่รันในคอนเทนเนอร์.

**วิธีที่ 2: ใช้ Docker Volume หรือ Shared Directory**

**1. สร้าง Docker Volume**

ขั้นตอนแรกคือการสร้าง Docker volume ซึ่งจะเป็นพื้นที่เก็บข้อมูลที่ใช้ร่วมกันระหว่างคอนเทนเนอร์:

bash

docker volume create shared-volume

**2. รันคอนเทนเนอร์และ mount volume**

ตอนนี้เราจะรันทั้งสองคอนเทนเนอร์ และทำการ mount volume ที่สร้างขึ้นไปยังตำแหน่งที่ต้องการในคอนเทนเนอร์ทั้งสอง:

**รัน docker1 และ mount volume:**

bash

docker run -d --name docker1 -v shared-volume:/shared-volume your\_image

docker run -d --name testreact-frontend-1 -v shared-volume:/shared-volume your\_image

**รัน docker2 และ mount volume:**

bash

docker run -d --name docker2 -v shared-volume:/shared-volume your\_image

ที่นี่, -v shared-volume:/shared-volume หมายถึงการ mount volume shared-volume ไปที่ /shared-volume ในคอนเทนเนอร์

**3. คัดลอกไฟล์จากคอนเทนเนอร์หนึ่งไปยัง volume**

เมื่อคอนเทนเนอร์ docker1 รันอยู่และ volume ถูก mount, คุณสามารถคัดลอกไฟล์จาก docker1 ไปยัง volume ได้:

bash

docker exec docker1 cp /path/to/file /shared-volume/tong.txt

**4. คัดลอกไฟล์จาก volume ไปยังคอนเทนเนอร์ docker2**

จากนั้น, คุณสามารถเข้าถึงไฟล์ใน docker2 ผ่าน shared volume ได้:

bash

docker exec docker2 cp /shared-volume/tong.txt /path/to/destination/in/docker2

**5. ผลลัพธ์**

ตอนนี้ไฟล์ tong.txt จะถูกคัดลอกจาก docker1 ไปยัง volume ที่แชร์กันและสามารถเข้าถึงได้จาก docker2 โดยไม่จำเป็นต้องคัดลอกจากโฮสต์ภายนอก.

**วิธีที่ 2: ใช้ Shared Directory (Directory บนโฮสต์)**

หากคุณต้องการใช้ directory ที่เป็น shared บนโฮสต์แทน volume:

**1. สร้าง shared directory บนโฮสต์**

สร้าง directory ที่จะใช้แชร์ข้อมูลระหว่างคอนเทนเนอร์:

bash

mkdir /path/to/shared-directory

**2. รันคอนเทนเนอร์และ mount shared directory**

ตอนนี้รันคอนเทนเนอร์ทั้งสองและ mount directory นี้เข้าไป:

**รัน docker1 และ mount shared directory:**

bash

docker run -d --name docker1 -v /path/to/shared-directory:/shared-dir your\_image

**รัน docker2 และ mount shared directory:**

bash

docker run -d --name docker2 -v /path/to/shared-directory:/shared-dir your\_image

**3. คัดลอกไฟล์จากคอนเทนเนอร์ docker1 ไปยัง shared directory**

bash

docker exec docker1 cp /path/to/file /shared-dir/tong.txt

**4. คัดลอกไฟล์จาก shared directory ไปยังคอนเทนเนอร์ docker2**

bash

docker exec docker2 cp /shared-dir/tong.txt /path/to/destination/in/docker2

**สรุป**

ทั้งวิธีที่ 1 (Volume) และวิธีที่ 2 (Shared directory) เป็นวิธีที่สะดวกในการทำให้คอนเทนเนอร์สองตัวสามารถแชร์ข้อมูลและคัดลอกไฟล์ระหว่างกันได้อย่างง่ายดาย โดยไม่ต้องใช้ขั้นตอนการคัดลอกจากโฮสต์ไปยังคอนเทนเนอร์.