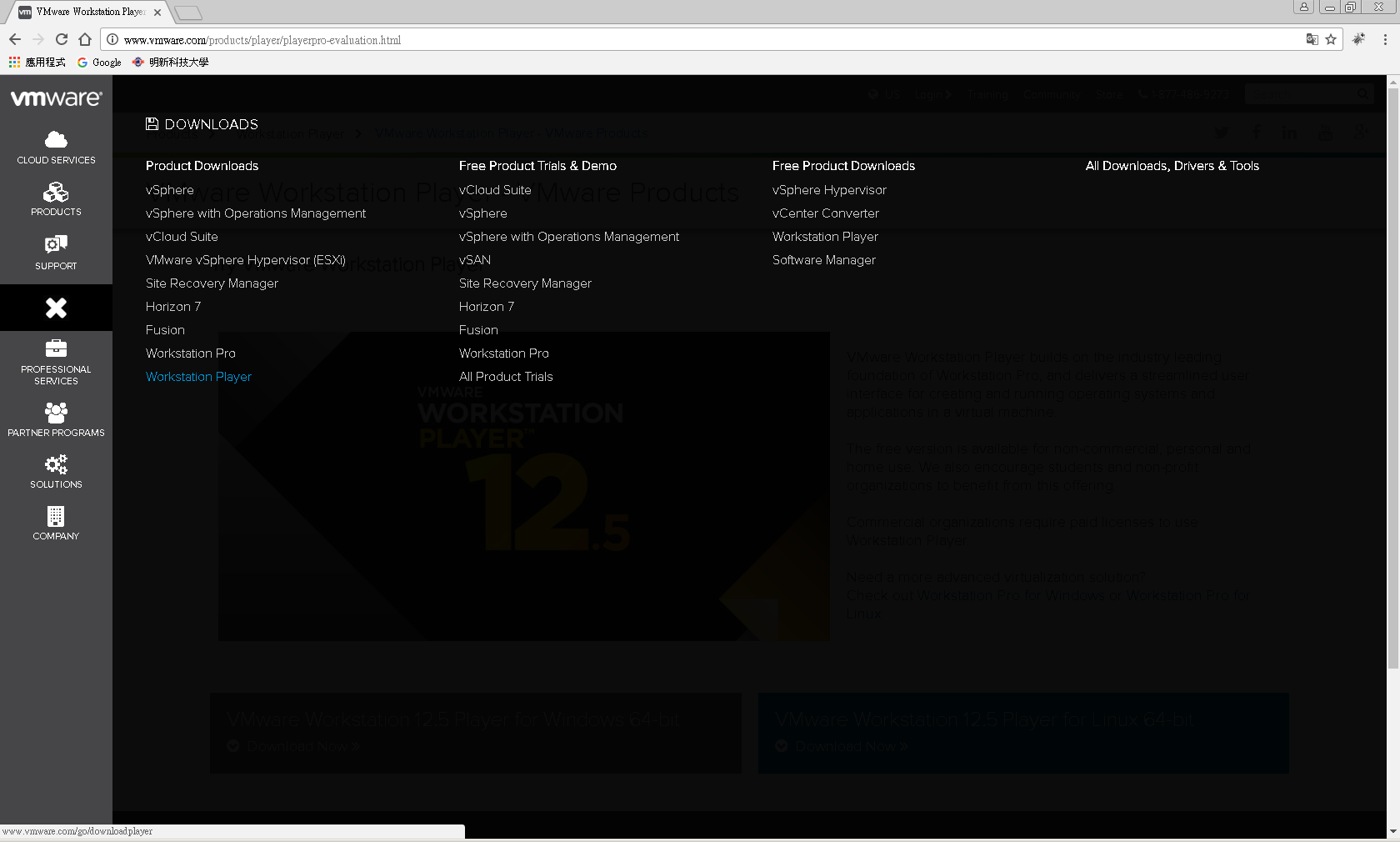
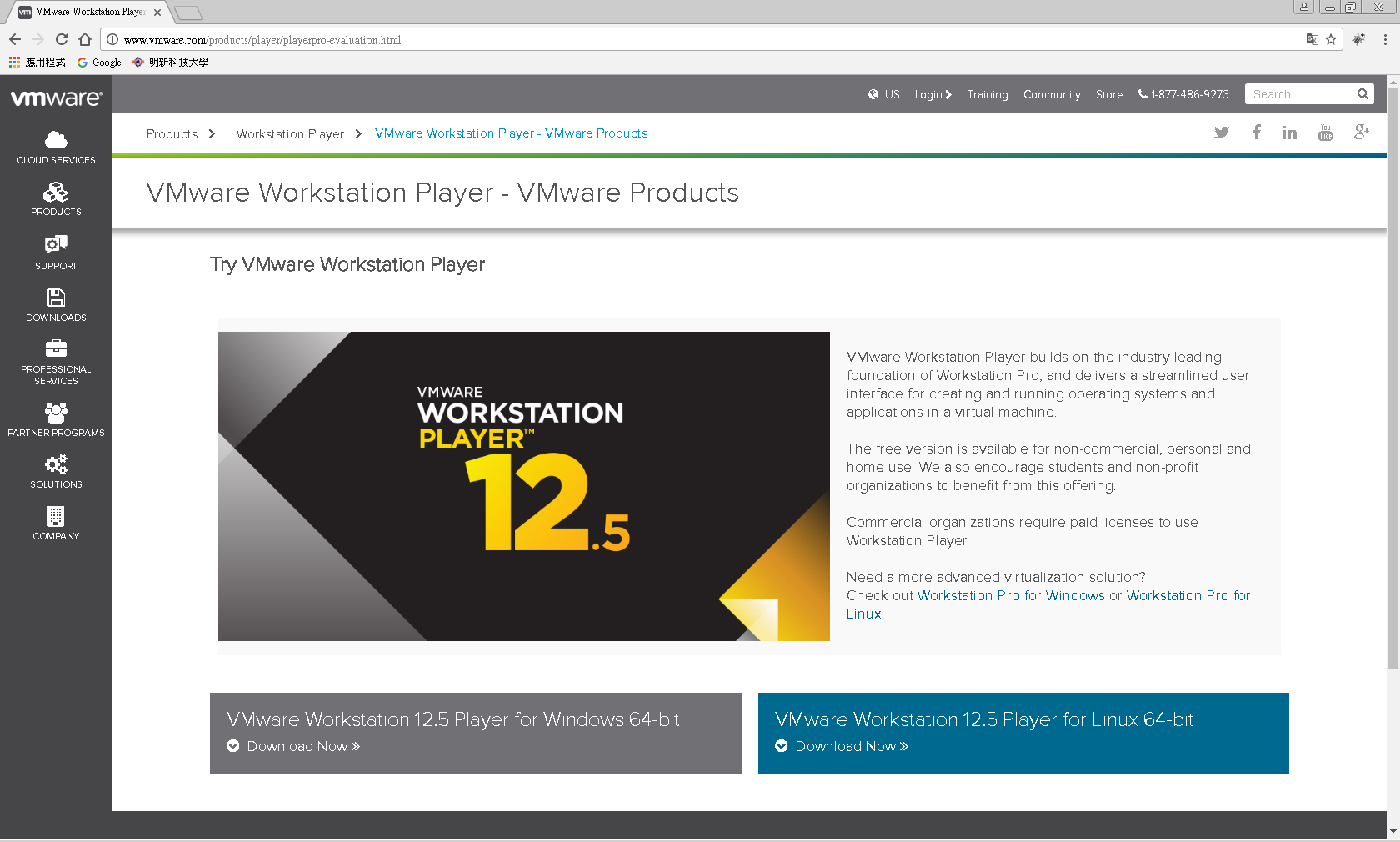
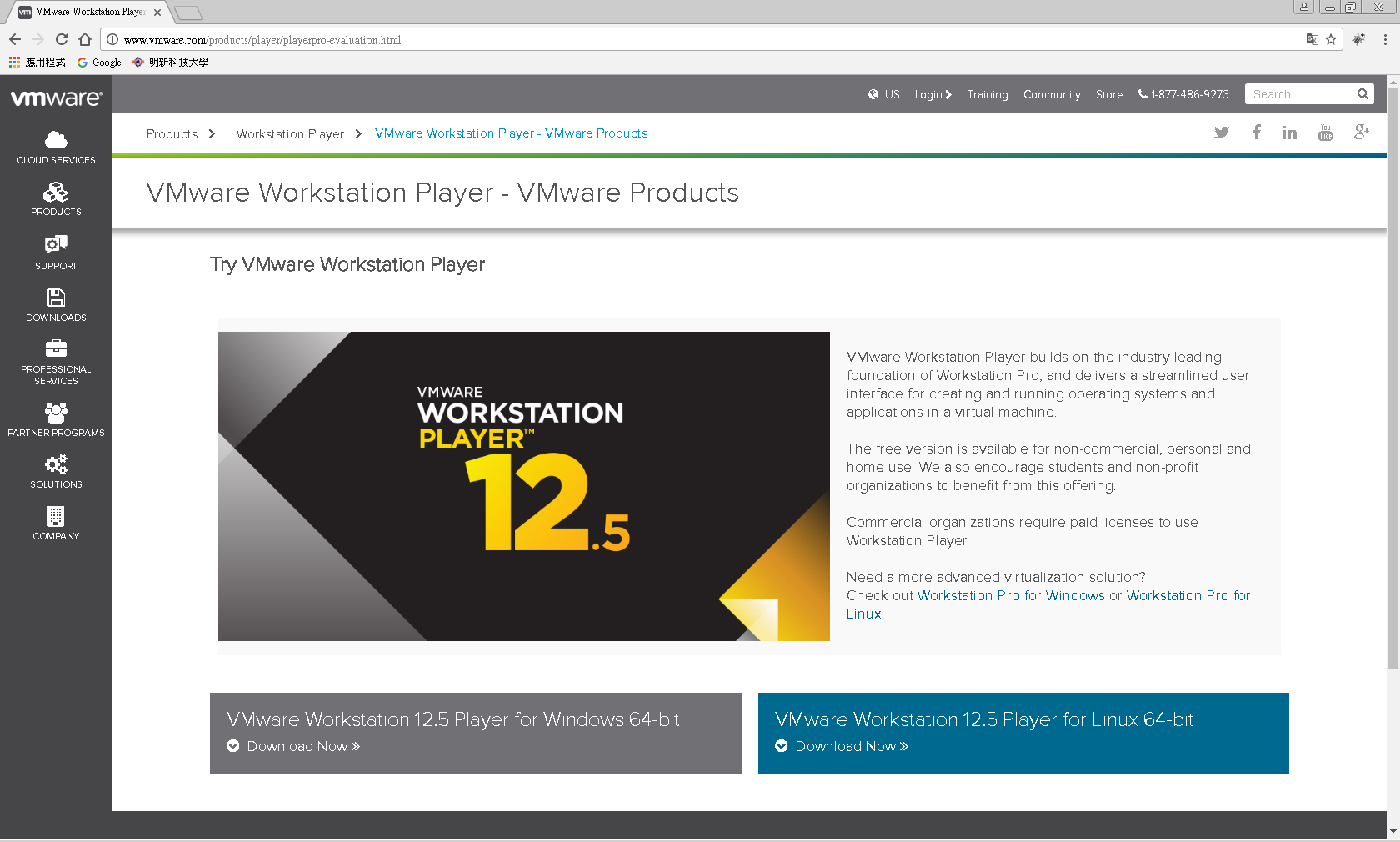
1. 安裝vmware workstation player虛擬機



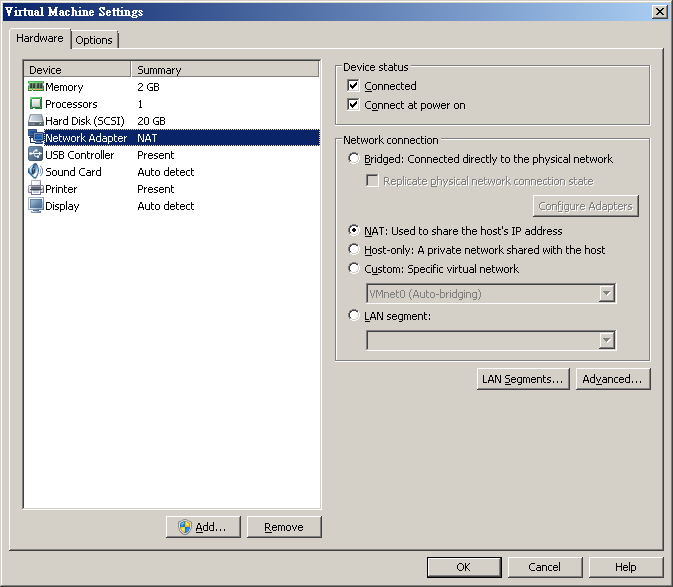




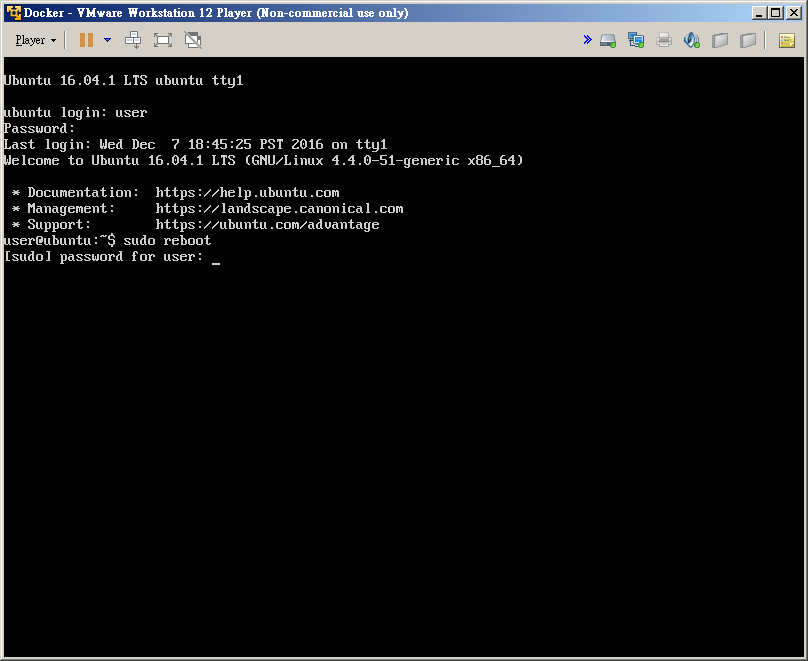
1. 在虛擬機的右上方，有個網路圖示，按滑鼠右鍵選擇”setting”



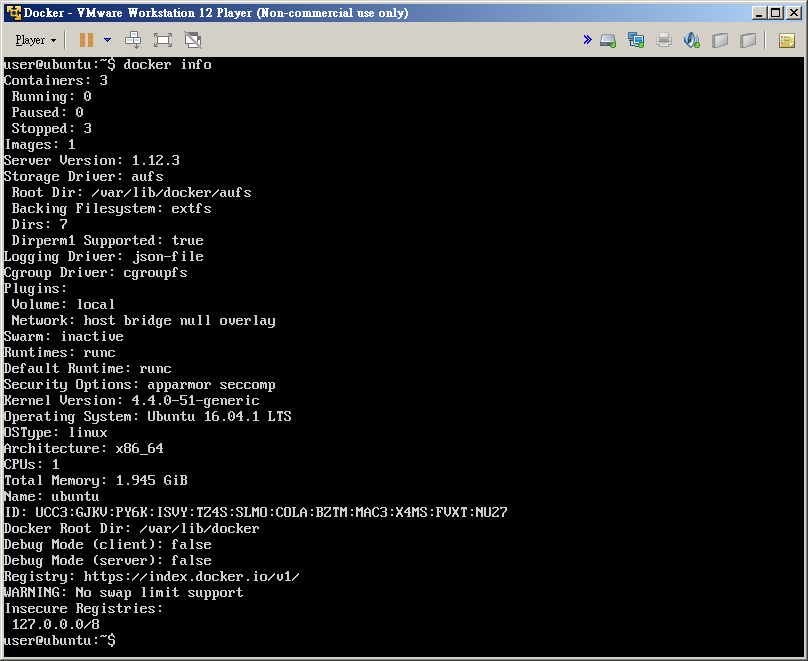
1. 把網路模式改成NAT



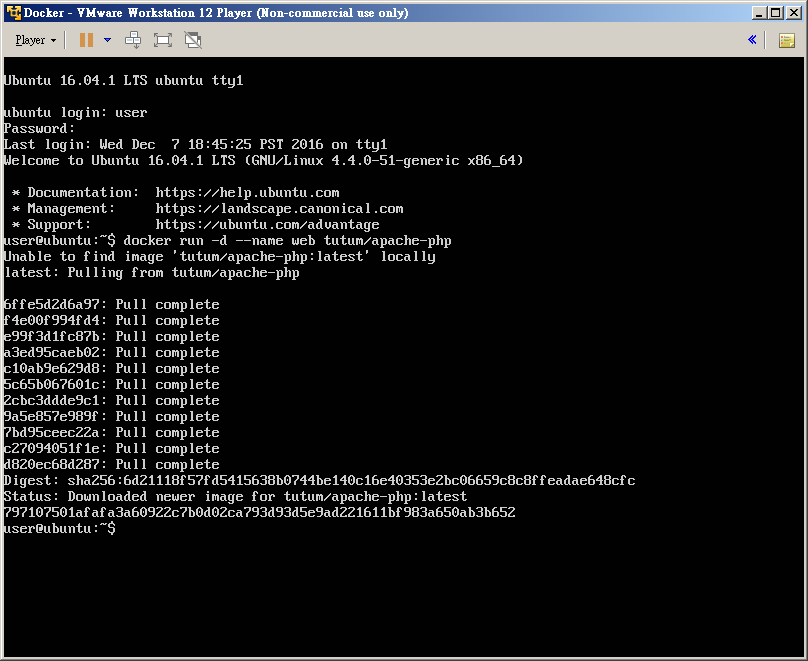
1. ID/PWD: user/user
2. 重新啟動sudo reboot (password:user)

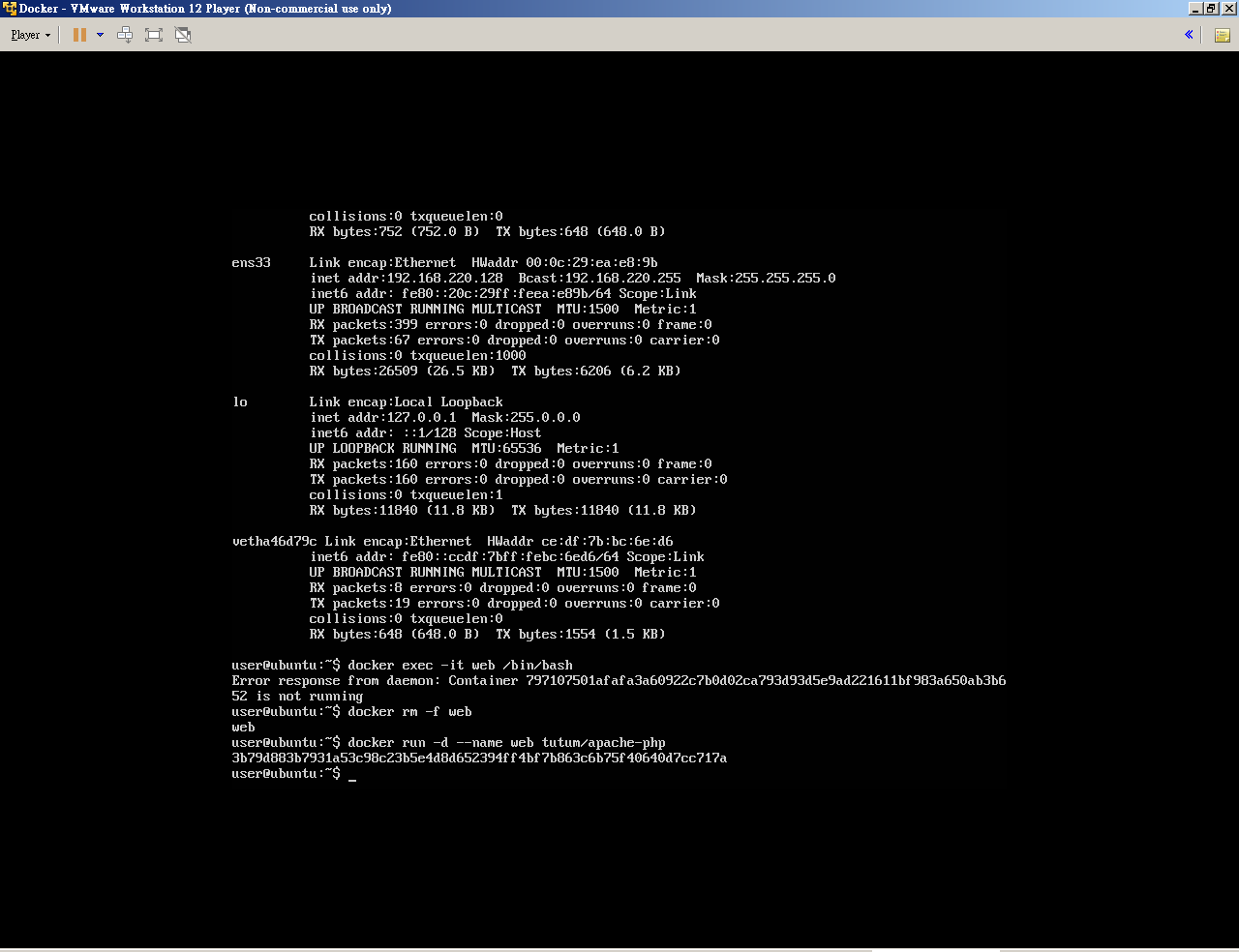


1. docker info (顯示docker資訊，有資訊代表連網成功)



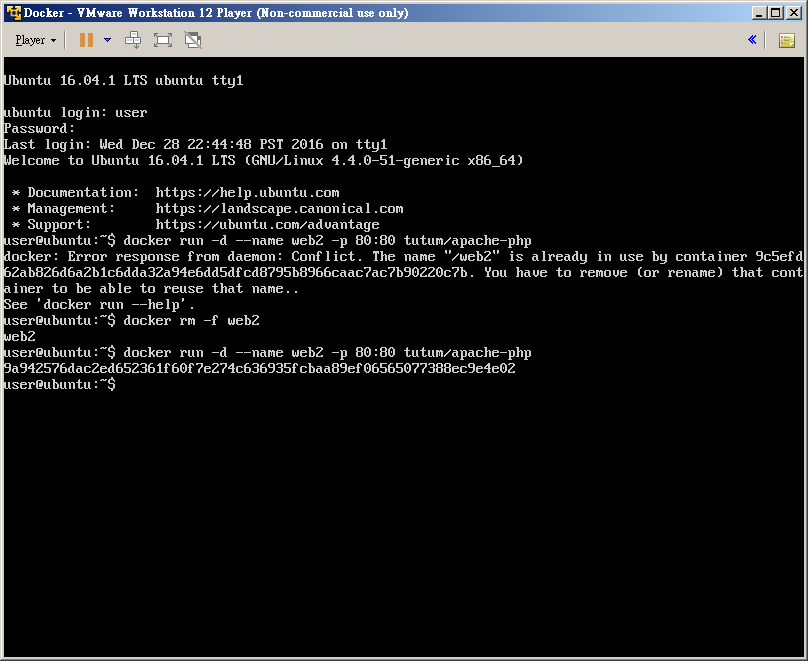
1. docker run -d --name **web**tutum/apache-php



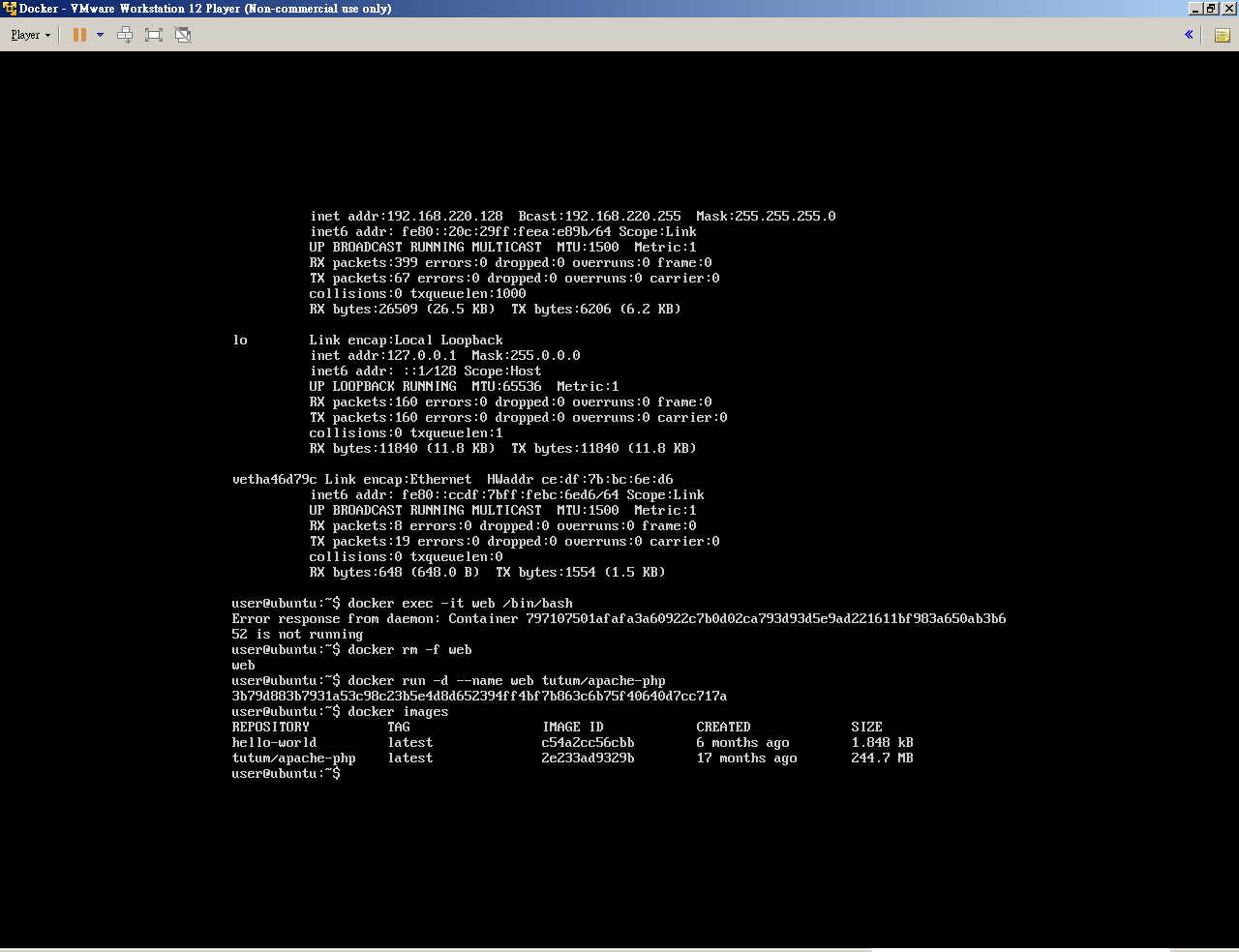


＊刪掉web之後再執行一次將會出現上面第二張的截圖!! (表示啟動web)

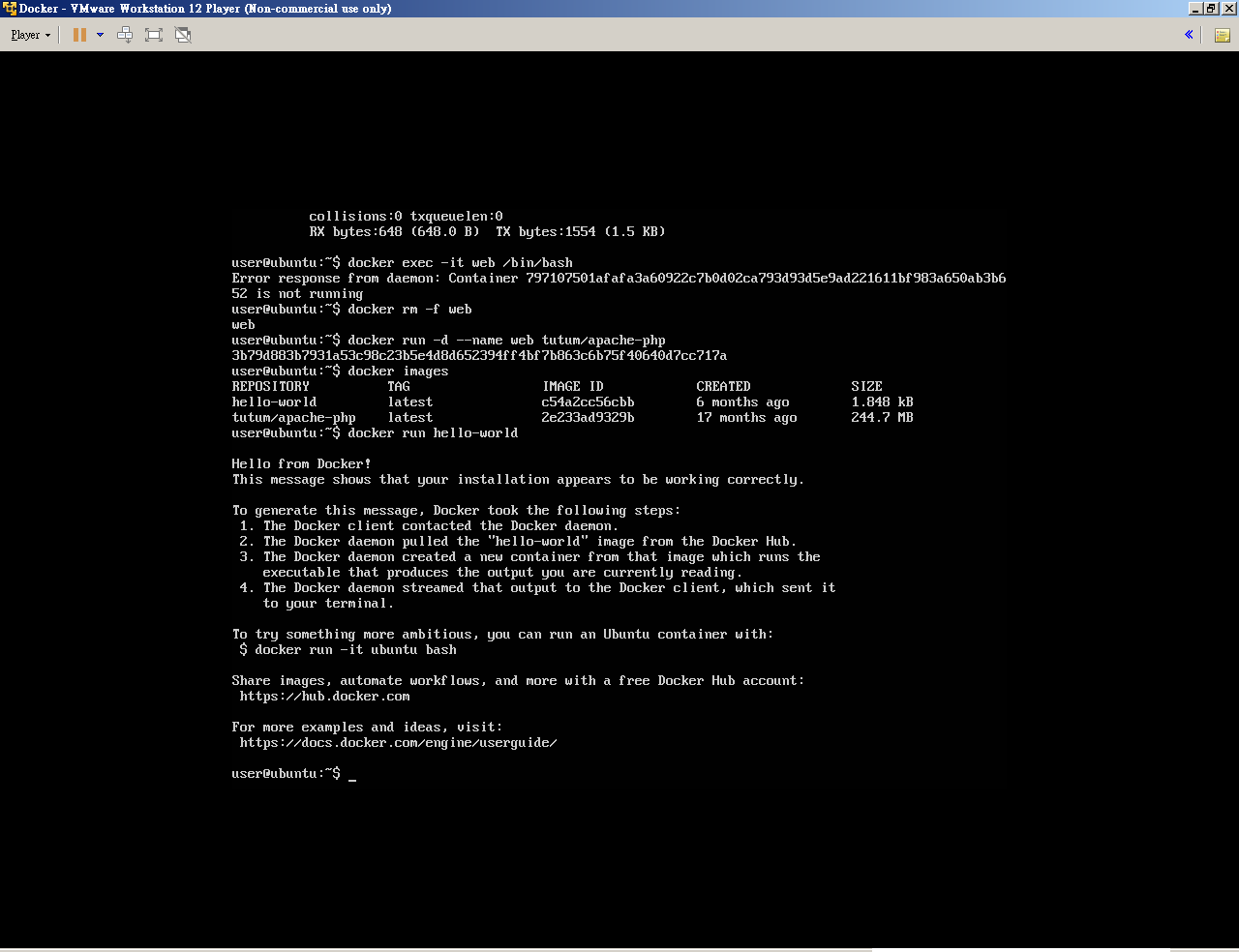
1. docker run -d --name **web2** -p 80:80 tutum/apache-php



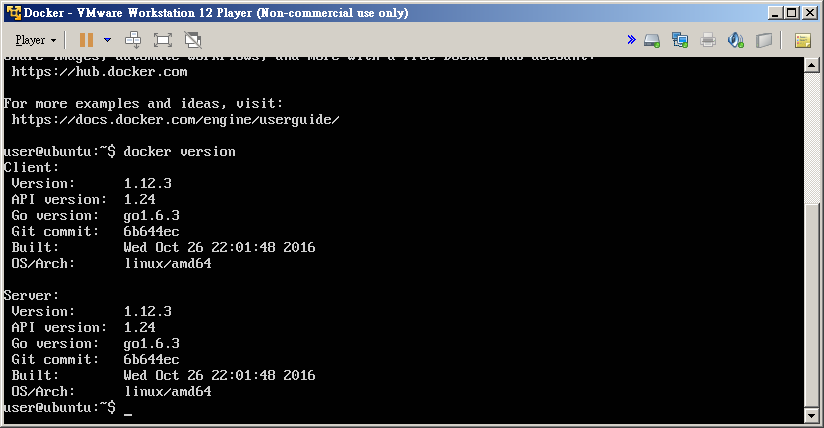
1. docker images



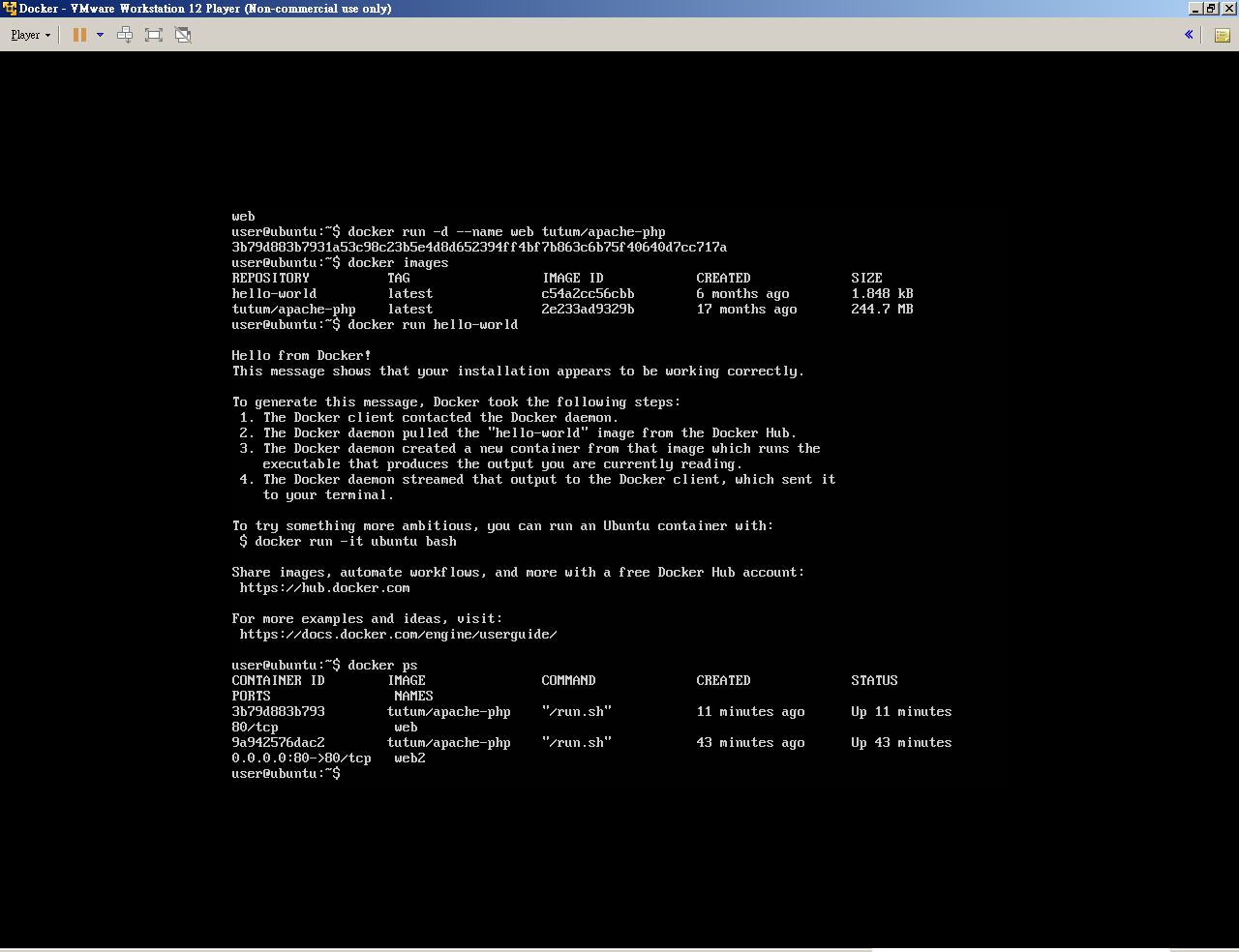
1. docker run hello-world



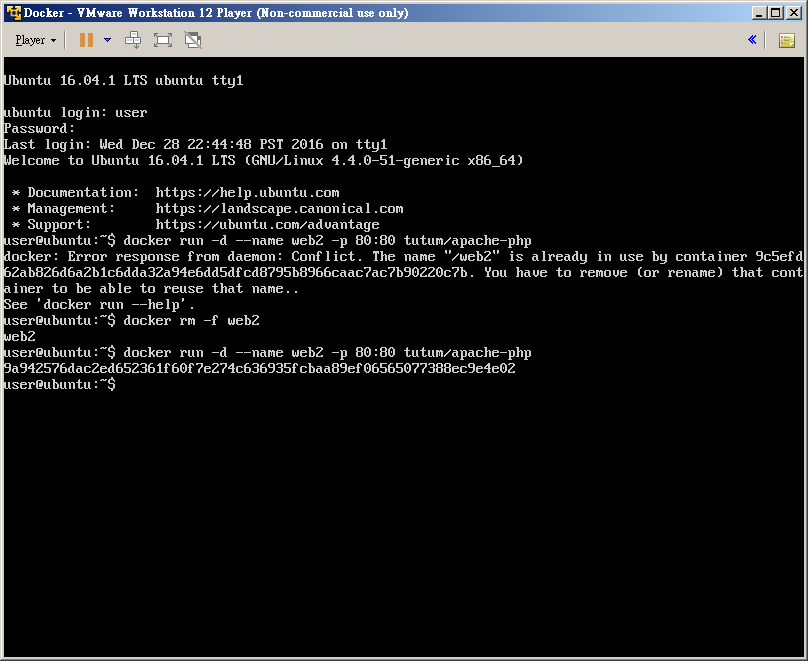
1. docker version查看版本



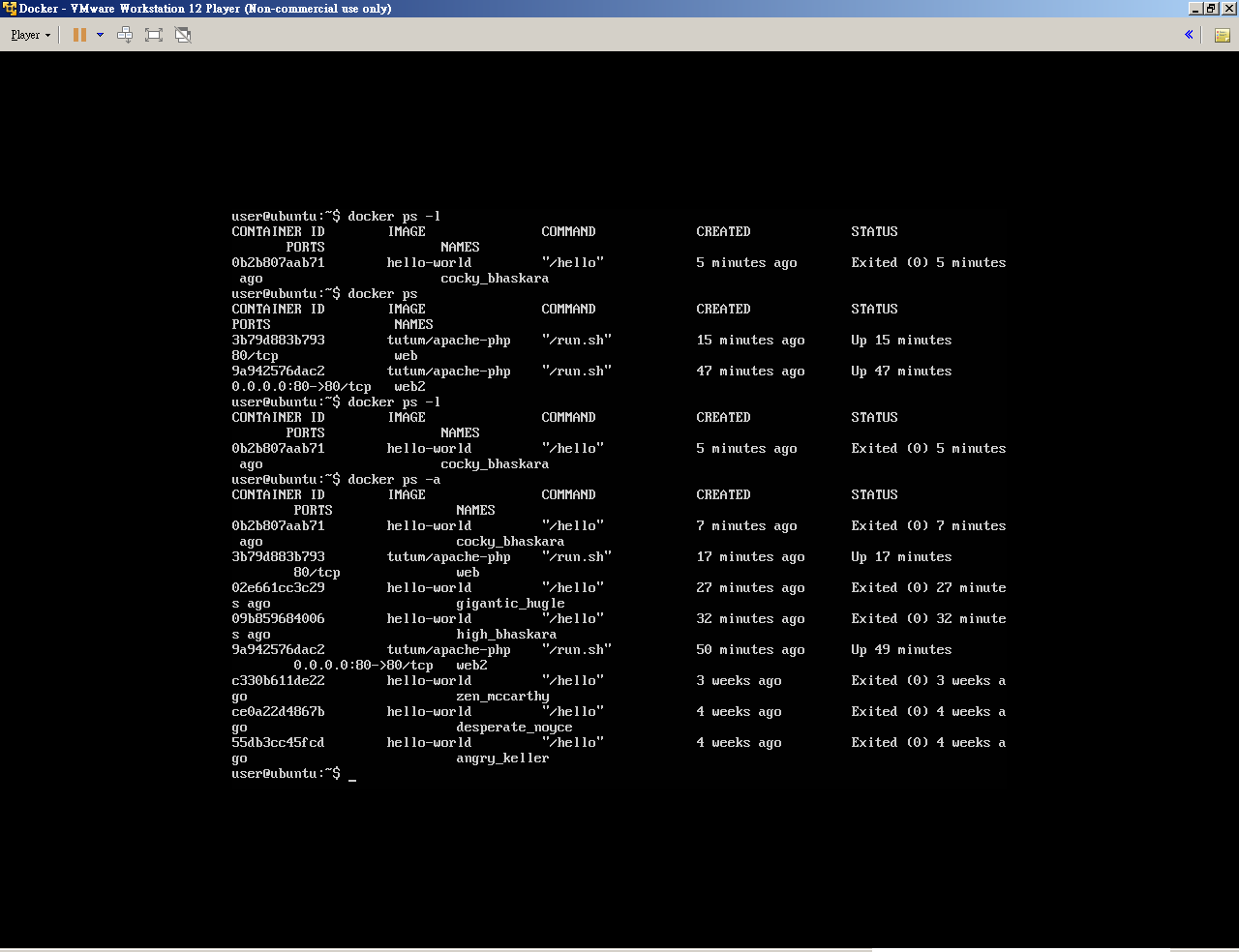
1. dockerps查看連接埠(?)



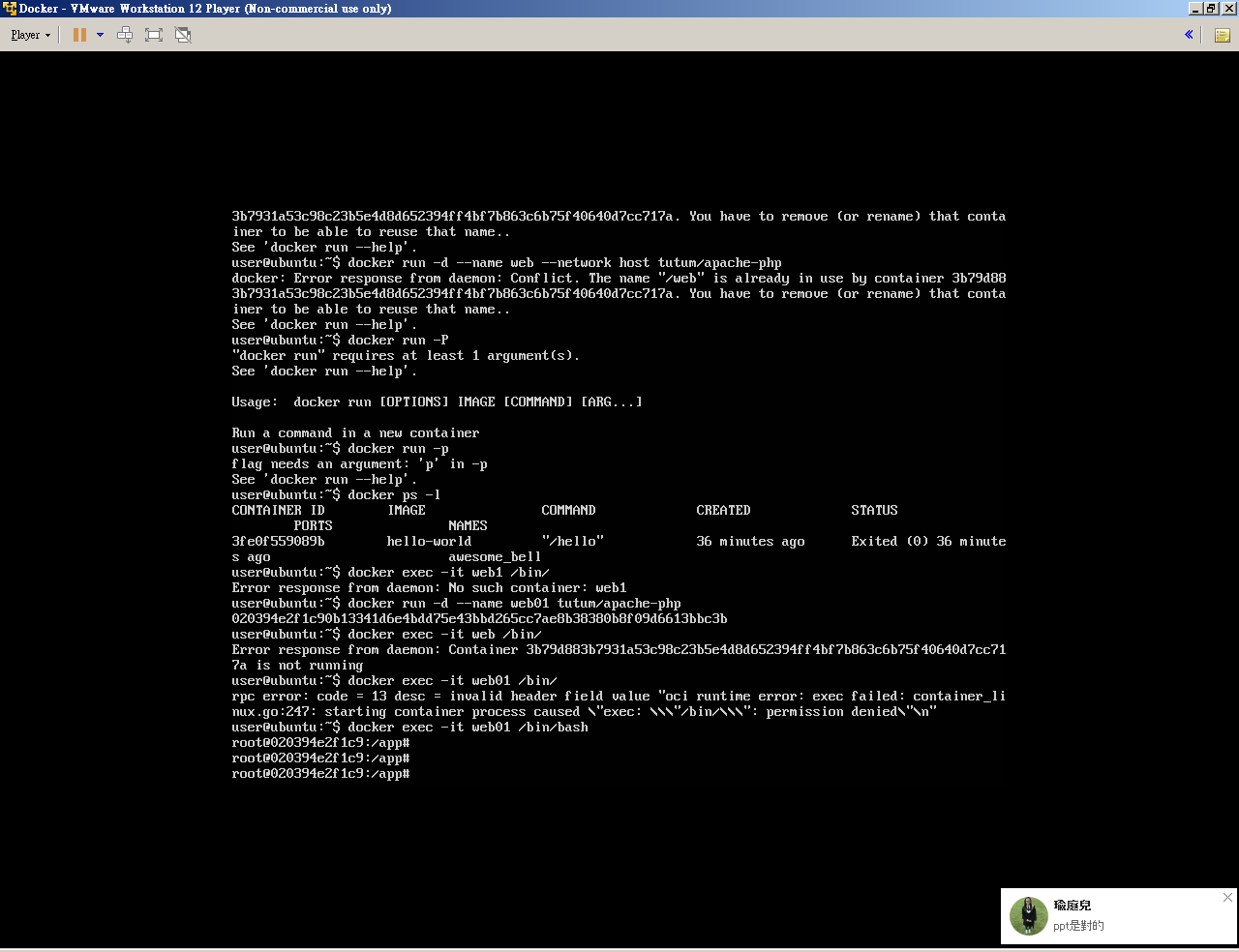
1. dockerrm–f web / rm web–f強制刪除



1. dockerps–a

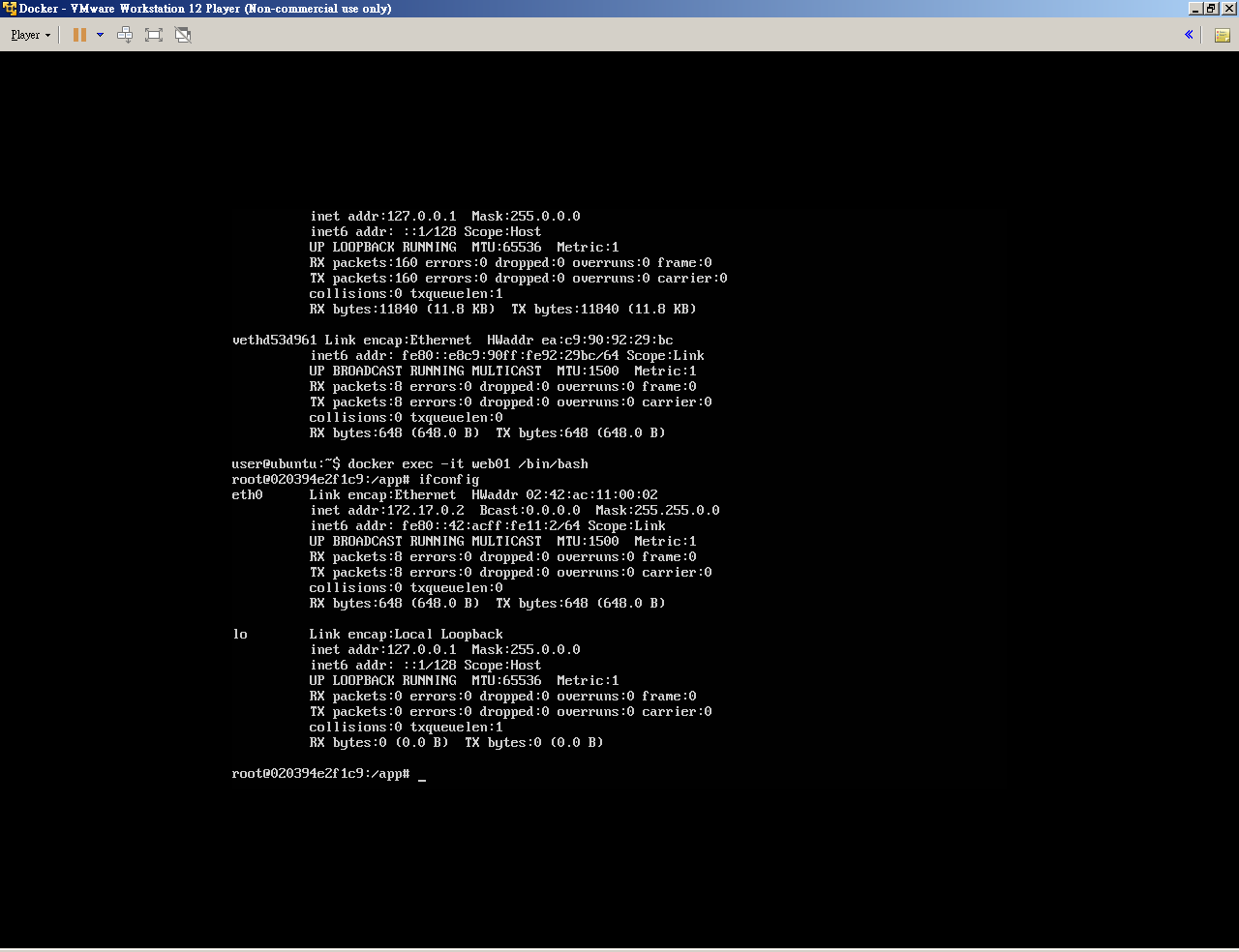


1. docker exec –it web /bin/bash(進入Container中觀察內部的網路卡)



C:\Users\Administrator\AppData\Local\LINE\Cache\tmp\1483003423672.jpg輸入exit 即可退出docker

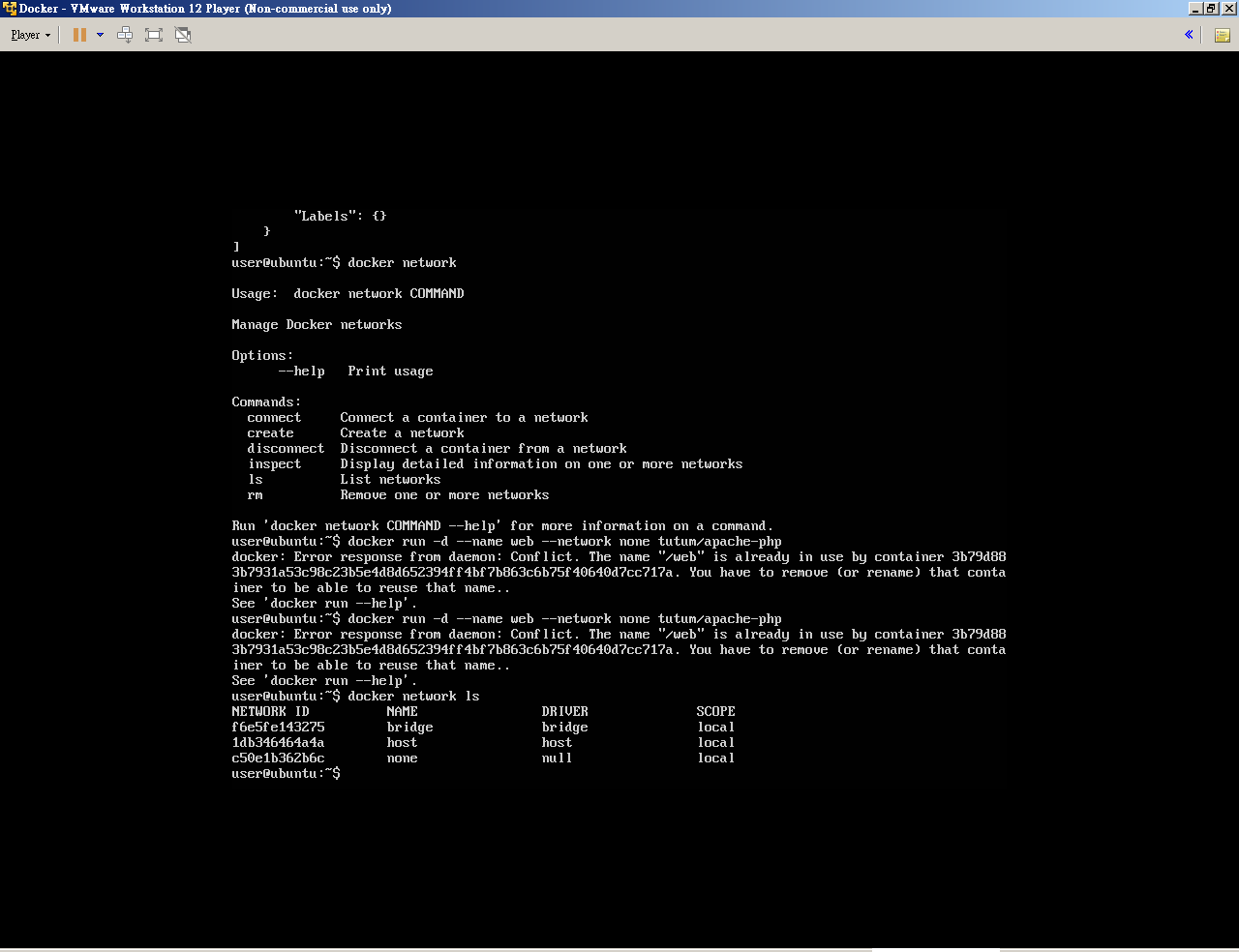
* ifconfig進入Container中觀察內部的網路卡



12/15 docker network筆記

1. docker network 看docker提供哪些網路連接模式

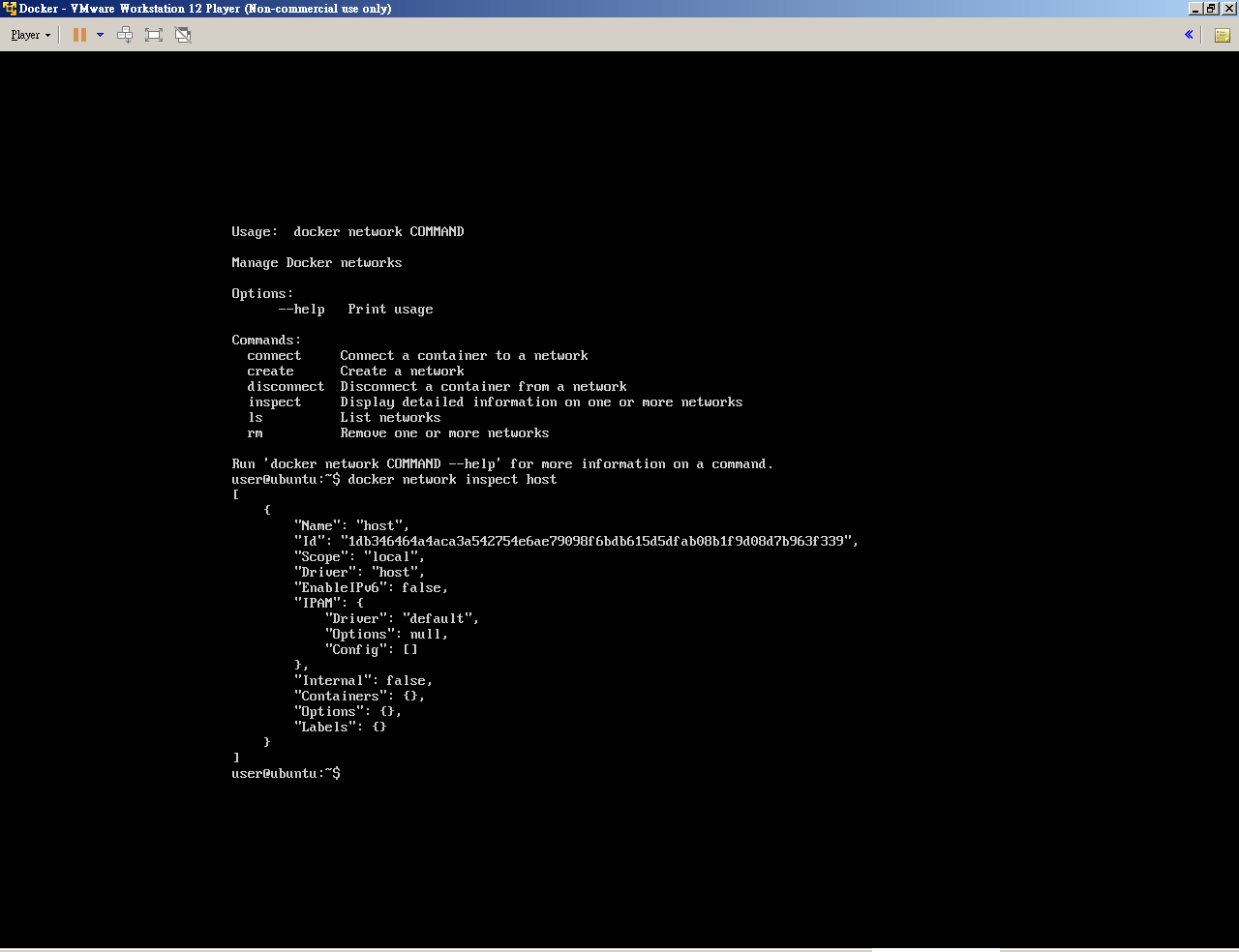
※查看目前有哪些橋接器，安裝時就預設裝了三個



1. docker network inspect host(or bridge or nat)

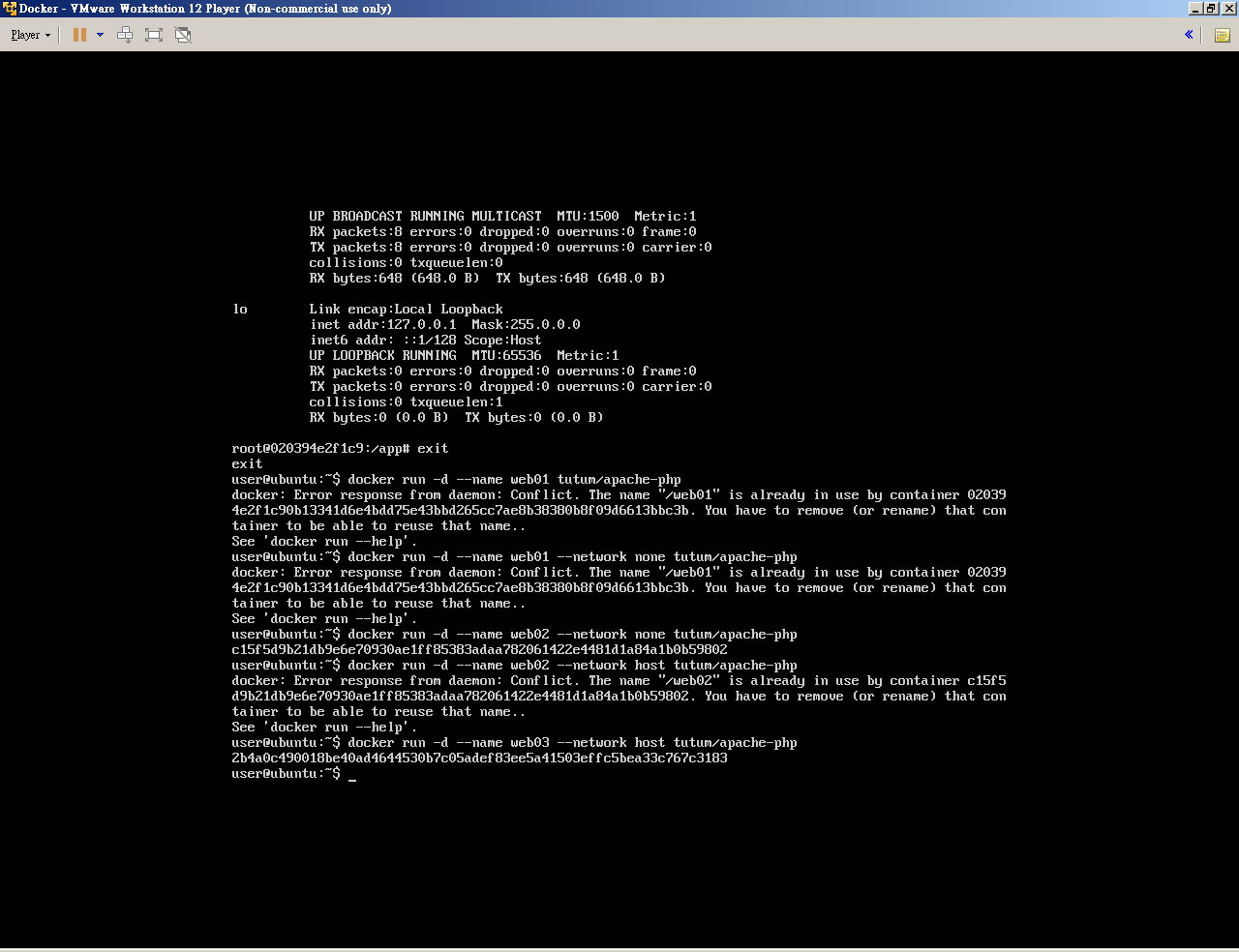
※查看目前橋接器的參數

docker network inspect <橋接器名稱>



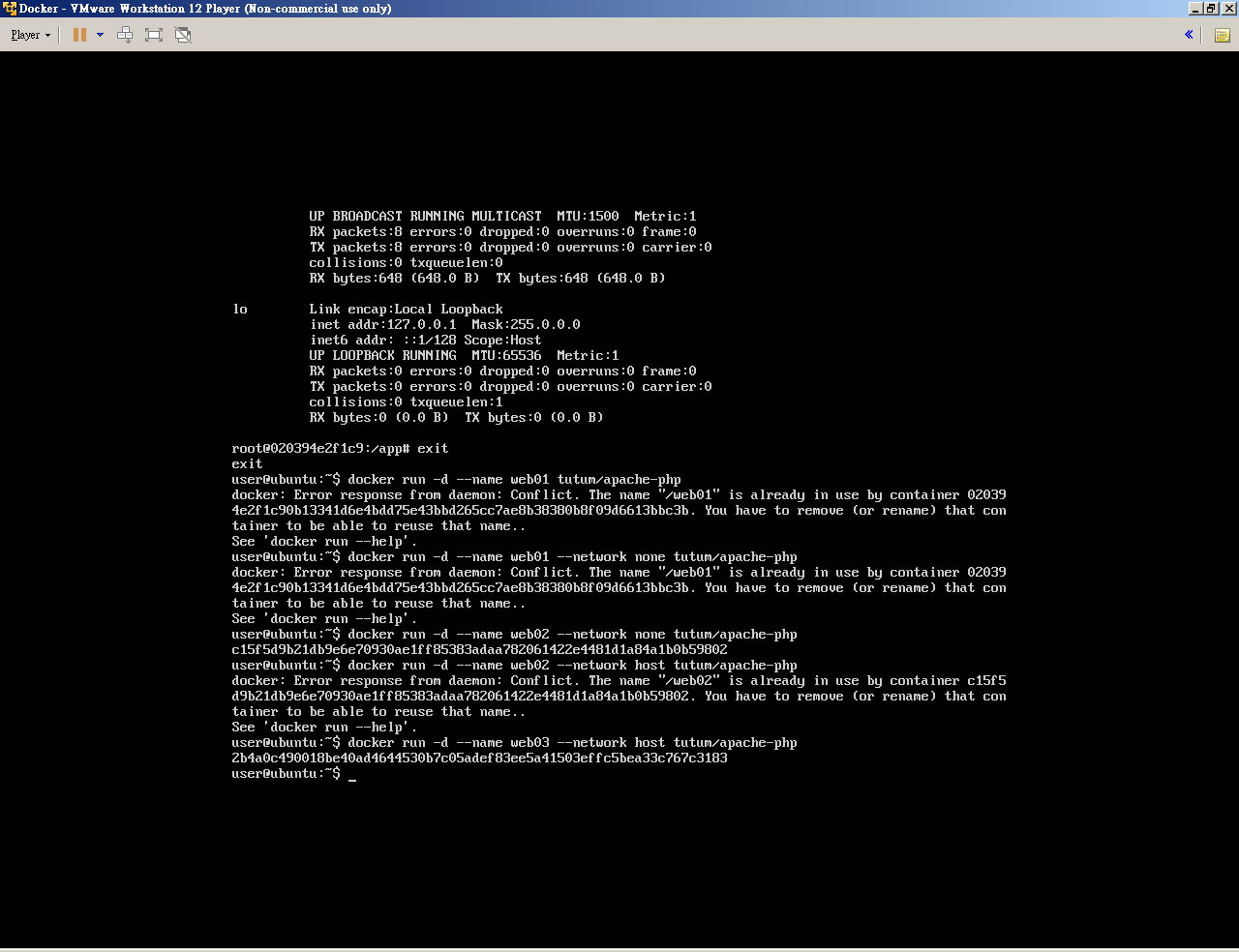
1. docker run –d --name web **--network none**tutum/apache-php

(container無網路設定)



1. docker run –d --name web **--network host**tutum/apache-php

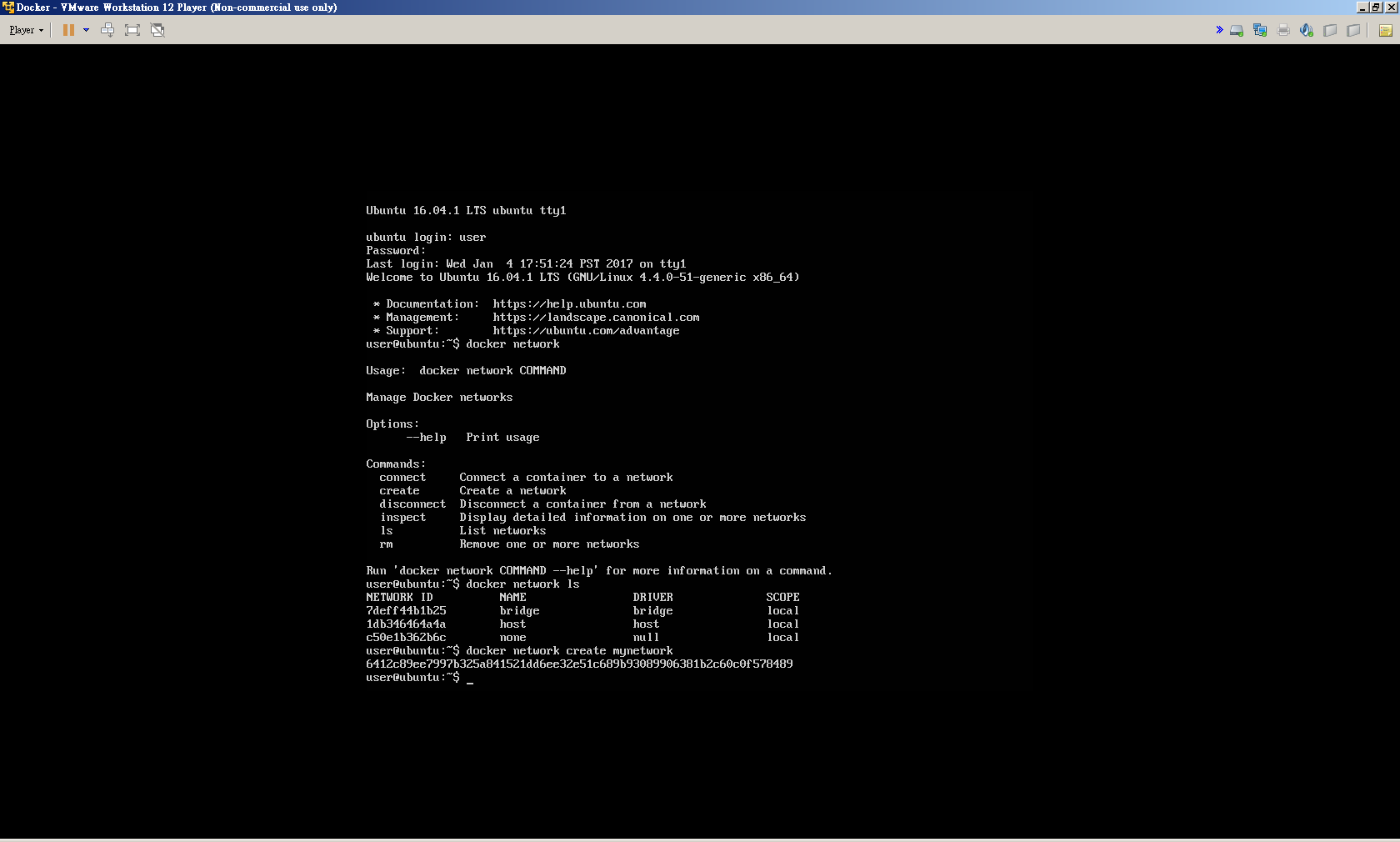
(container host設定，可直接由外部連接到container的第一種方式，不建議)



此時Container就連至host，表示接管了主系統的所有ip及網路設定

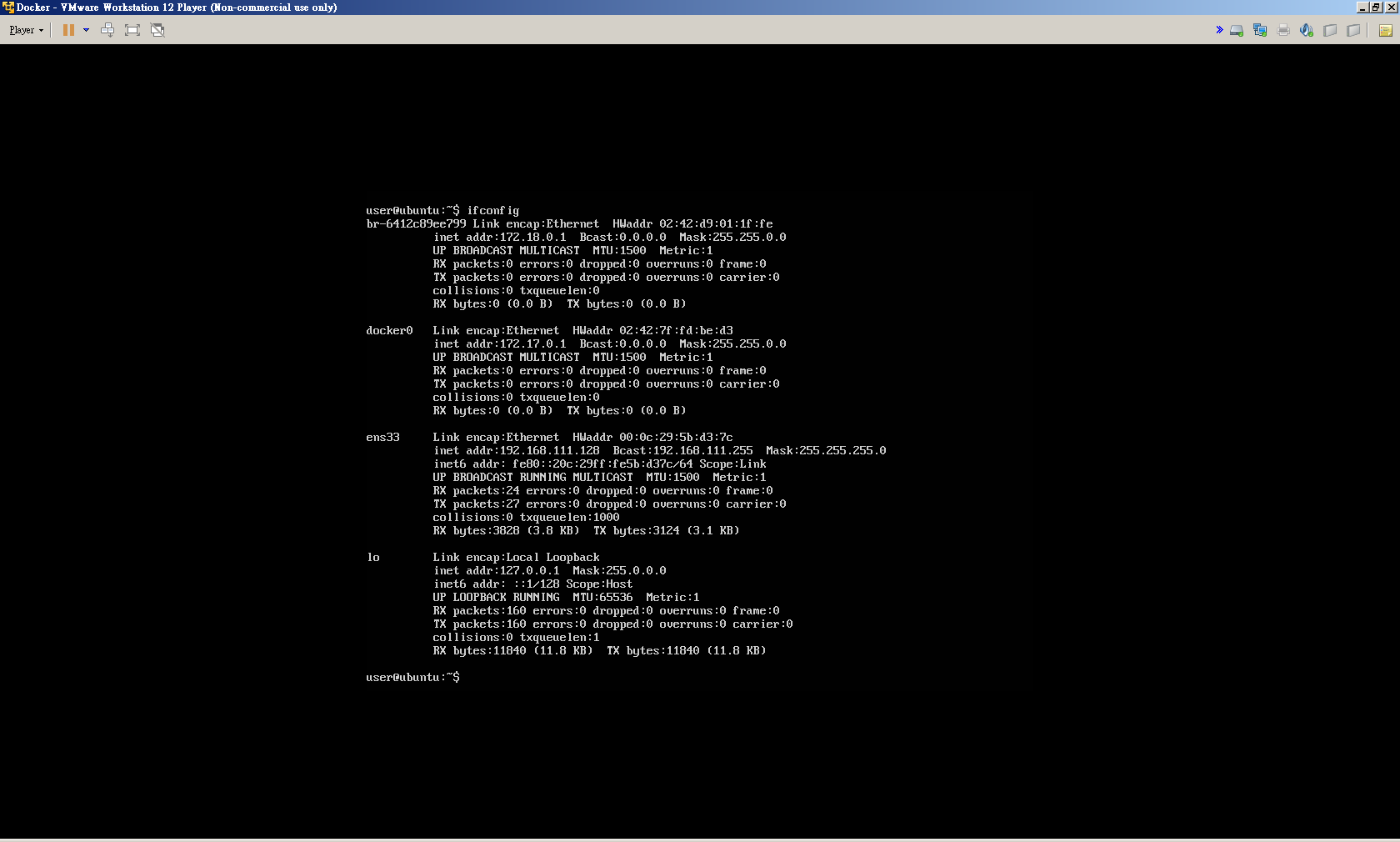
※自訂網路

1.docker network create mynetwork (自訂網路給一群Container用)

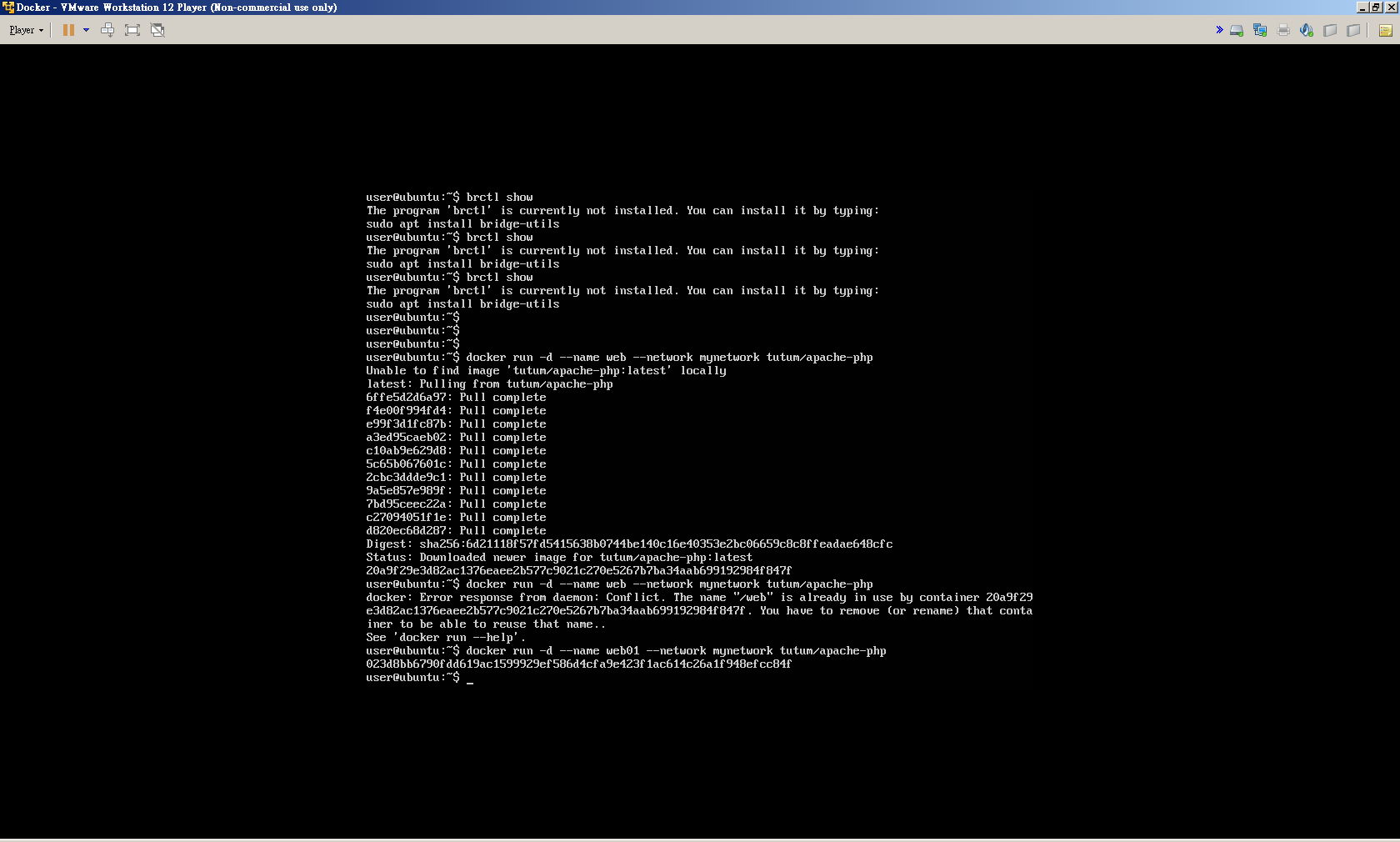


2.brctl show(來查看橋接器和連接埠連接訊息)

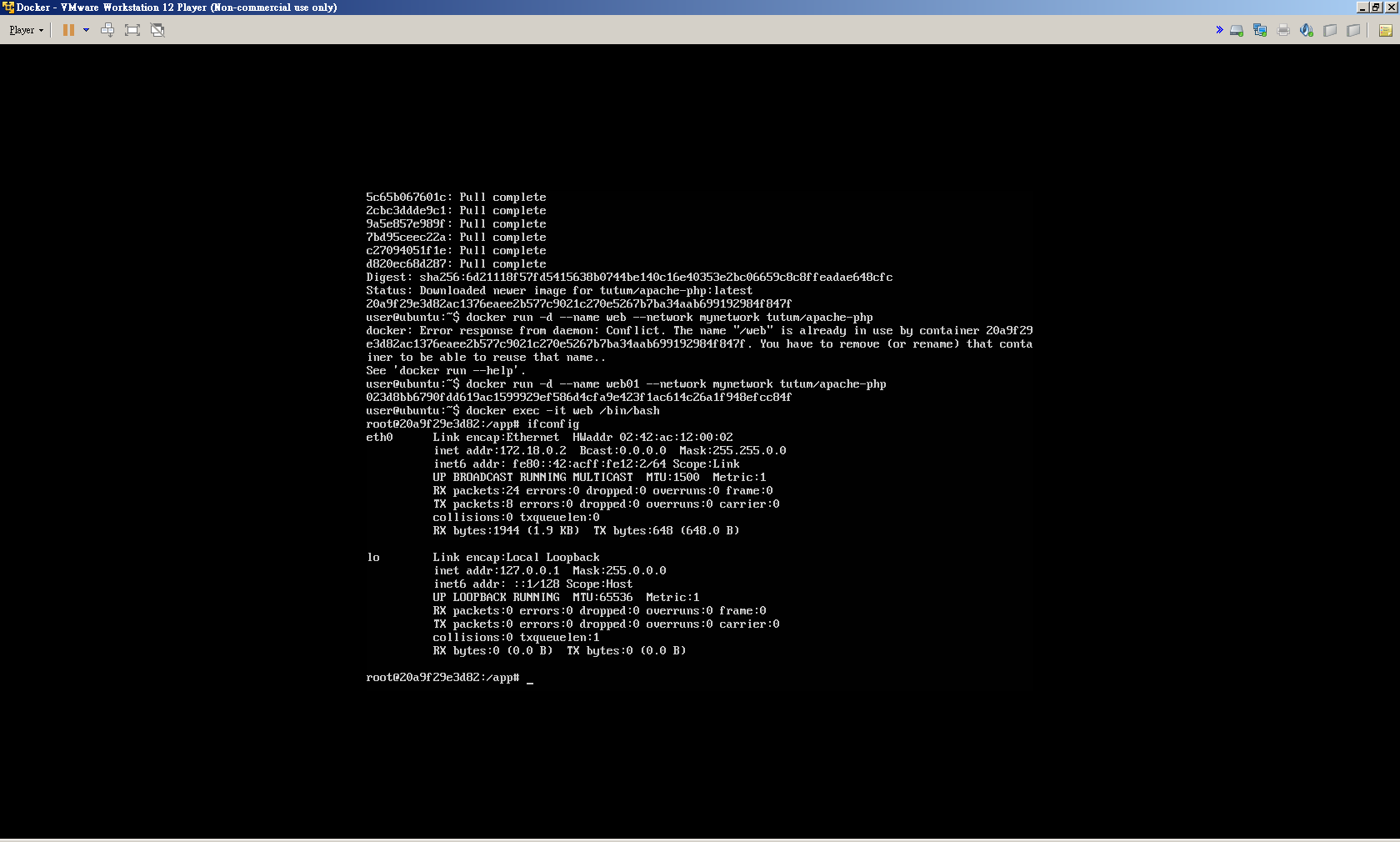
3. Ifconfig (查看目前建立的網路連線)



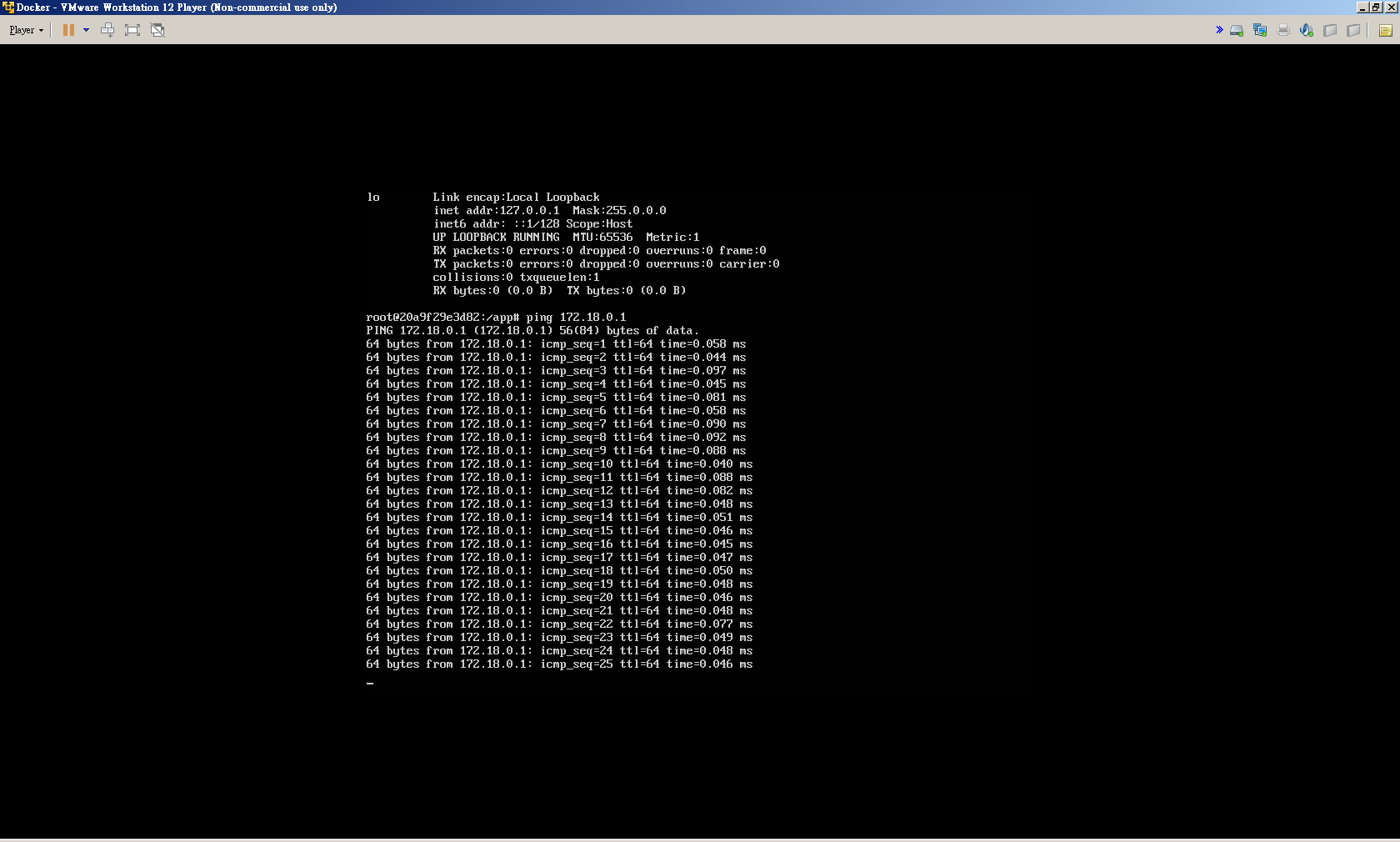
4.啟動Container時連上這個網路



5.docker exec –it web /bin/bash ; ifconfig (查看Container的網路連線)

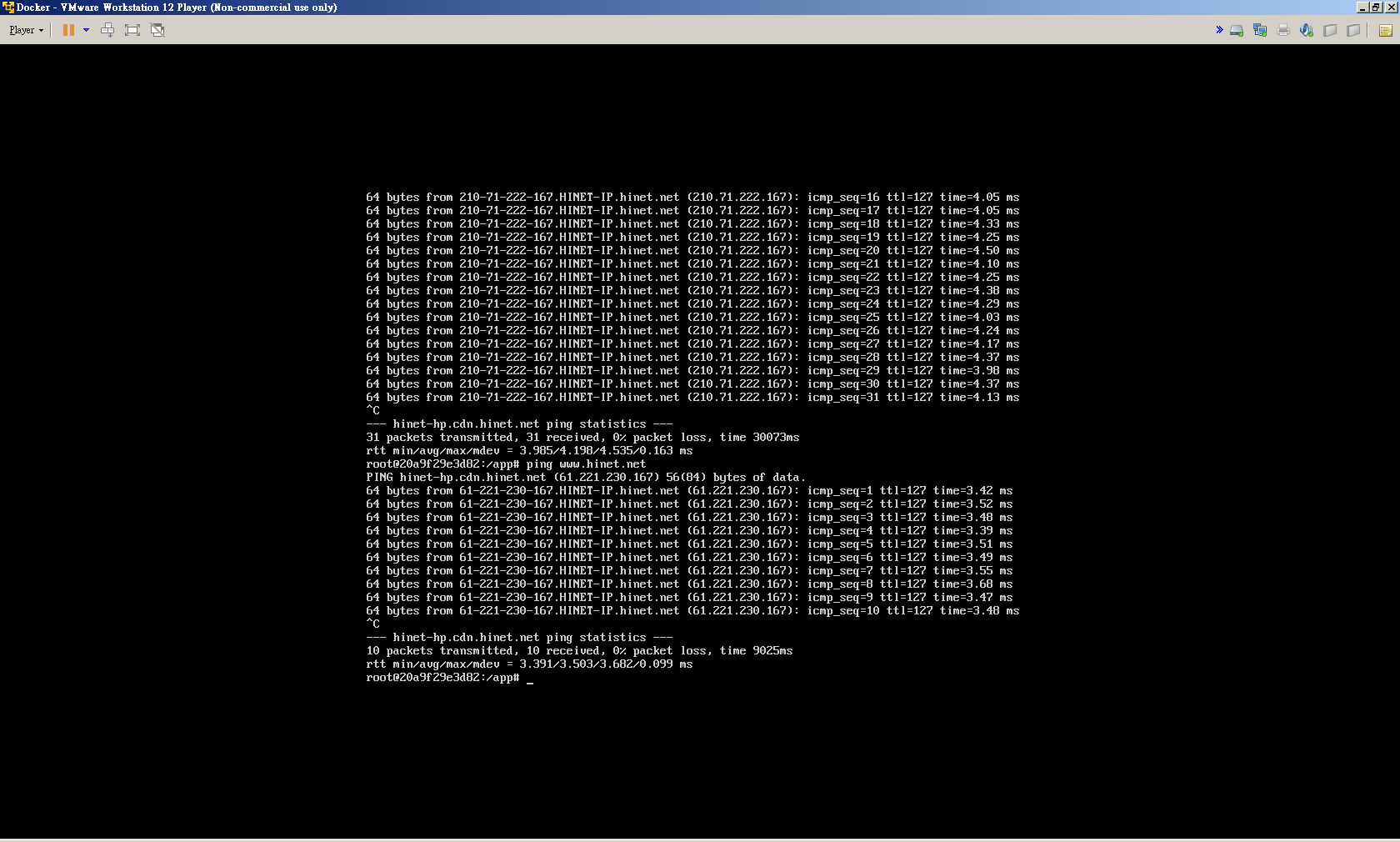


6.可連至主系統(ping 172.18.0.1)



按Ctrl+C停止傳送回覆訊息

7.預設可橋接至公網



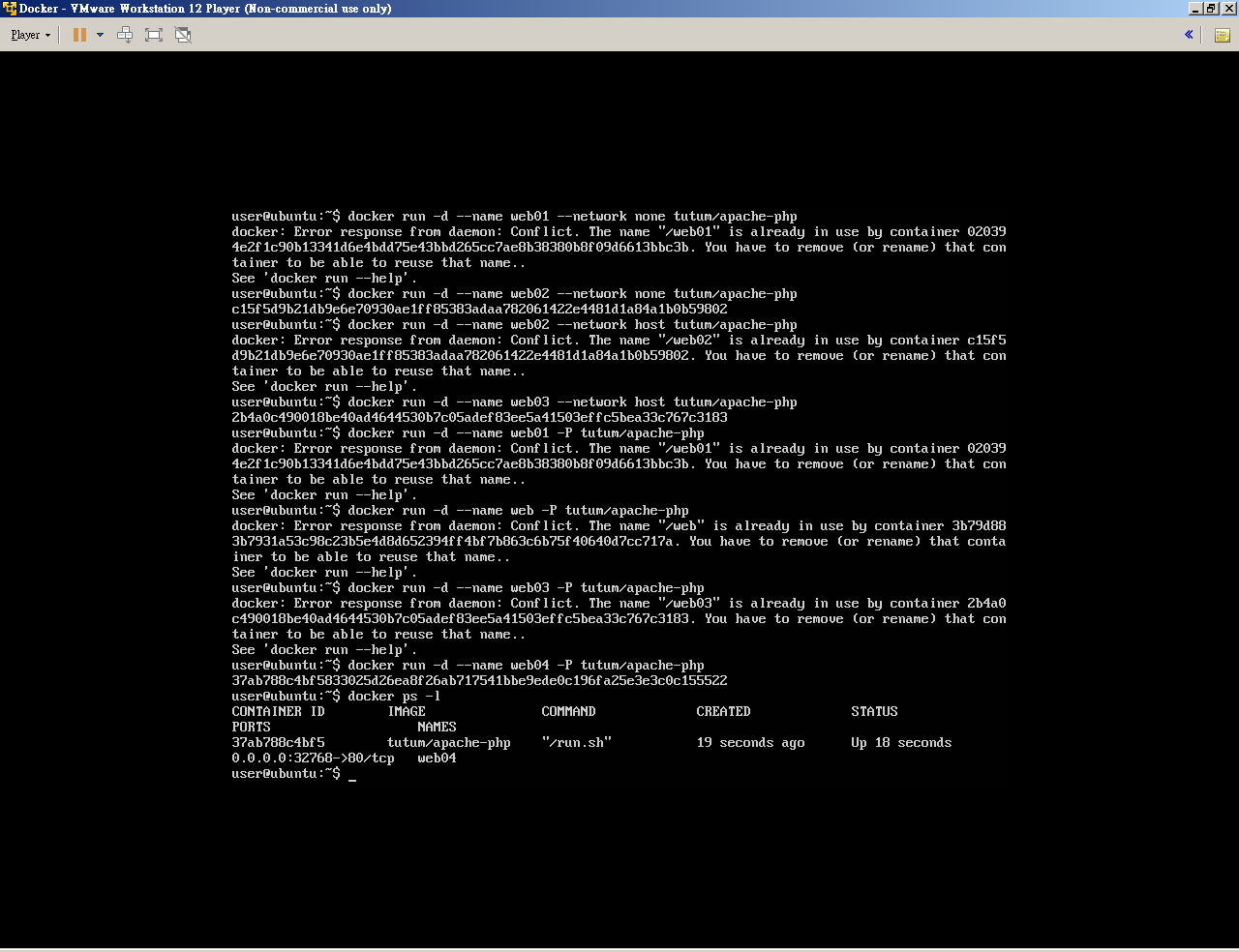
按Ctrl+C停止傳送回覆訊息

8.docker network connect bridge web ; ifconfig (將現存的Container連上其他網路)

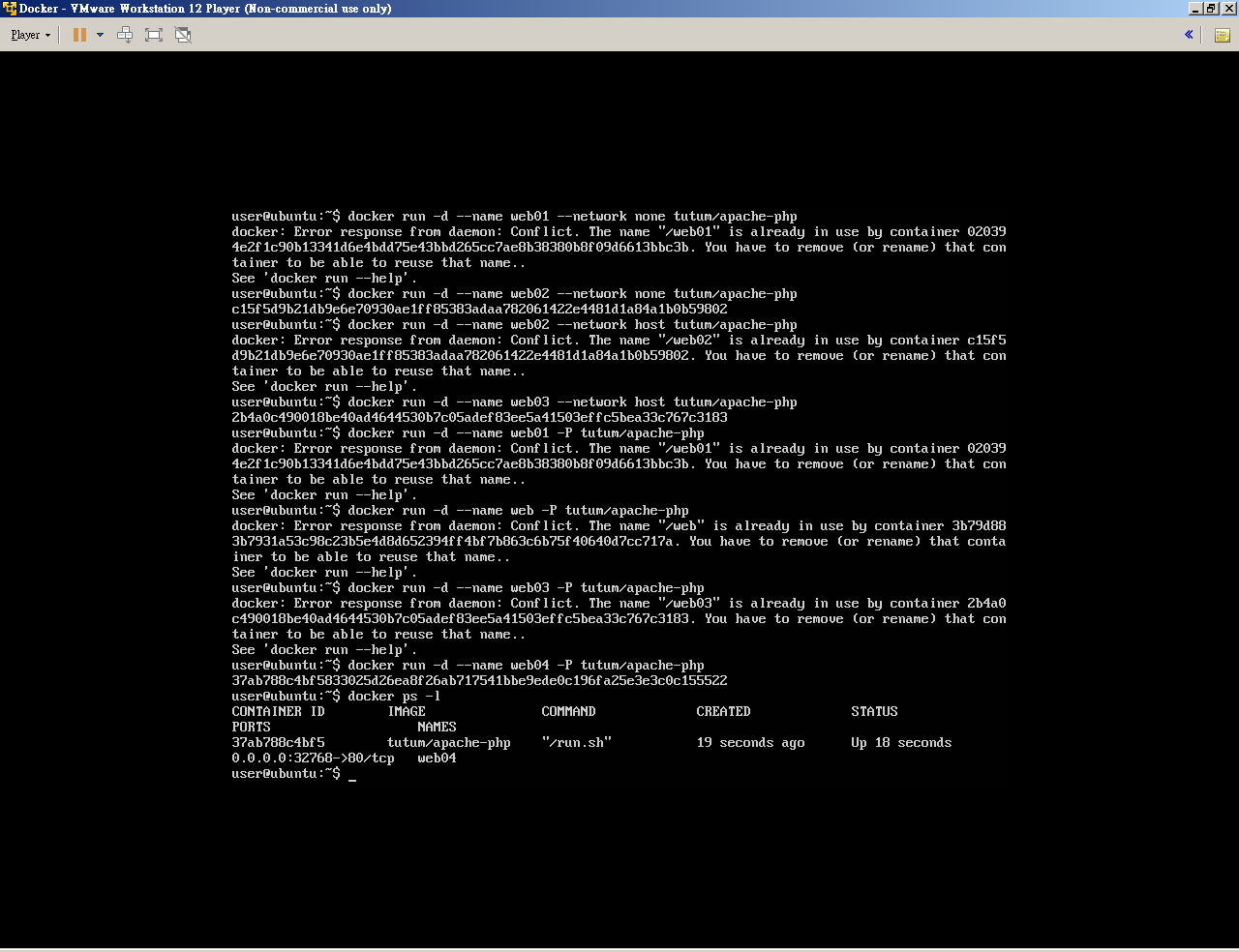
※通訊埠映射:

(由外部連接到container的第二種方式，新增網路介面來對應，較複雜)

docker run –P (隨機映射一個連接埠到內部容器開放的網路連接阜)

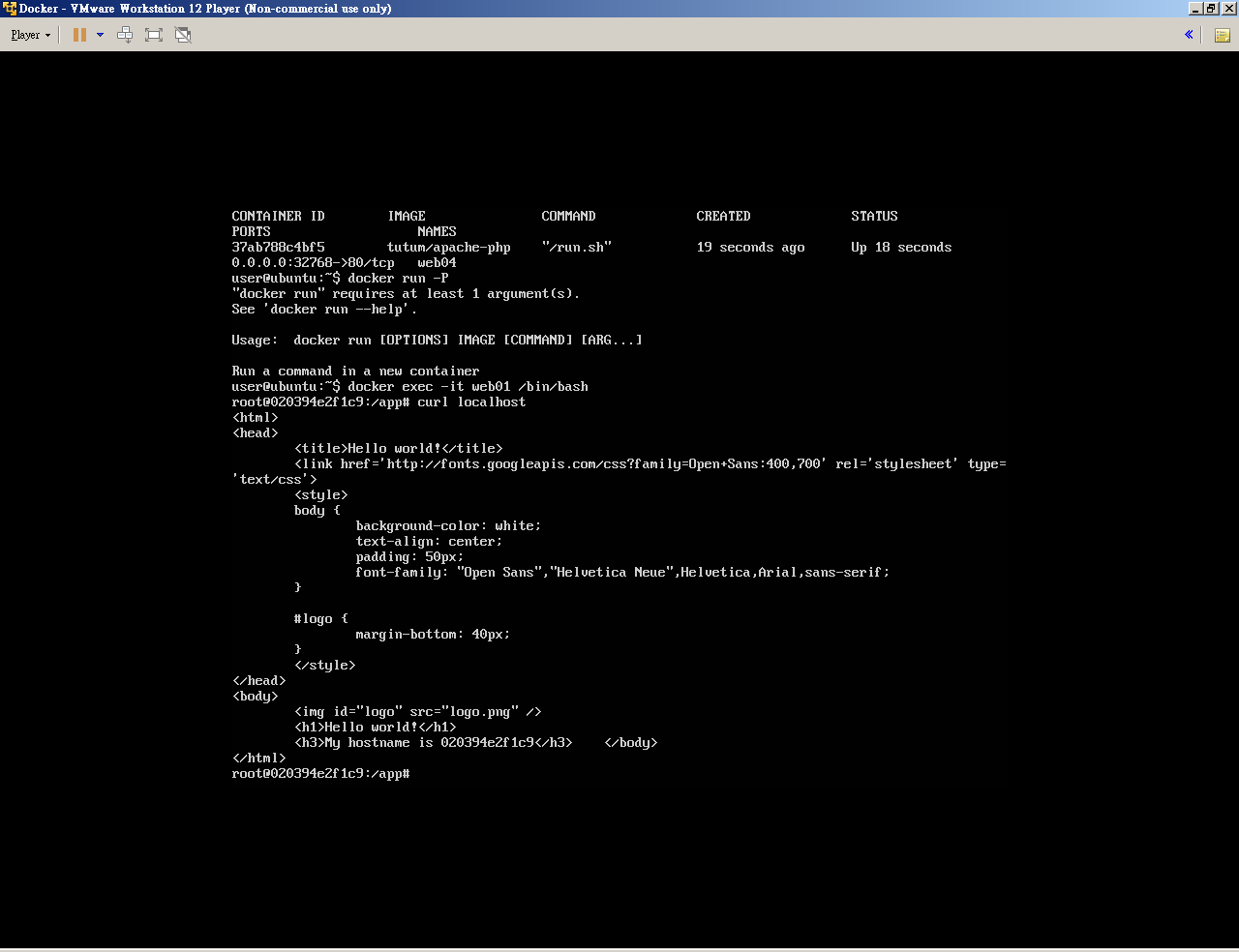


1. dockerps–l (觀察哪一個通訊埠)



1. 進入到container內部存取，打curl localhost測試

docker exec –it web /bin/bash ; curl localhost



* docker port <container> (看<container>用了哪些內部port及port對應)

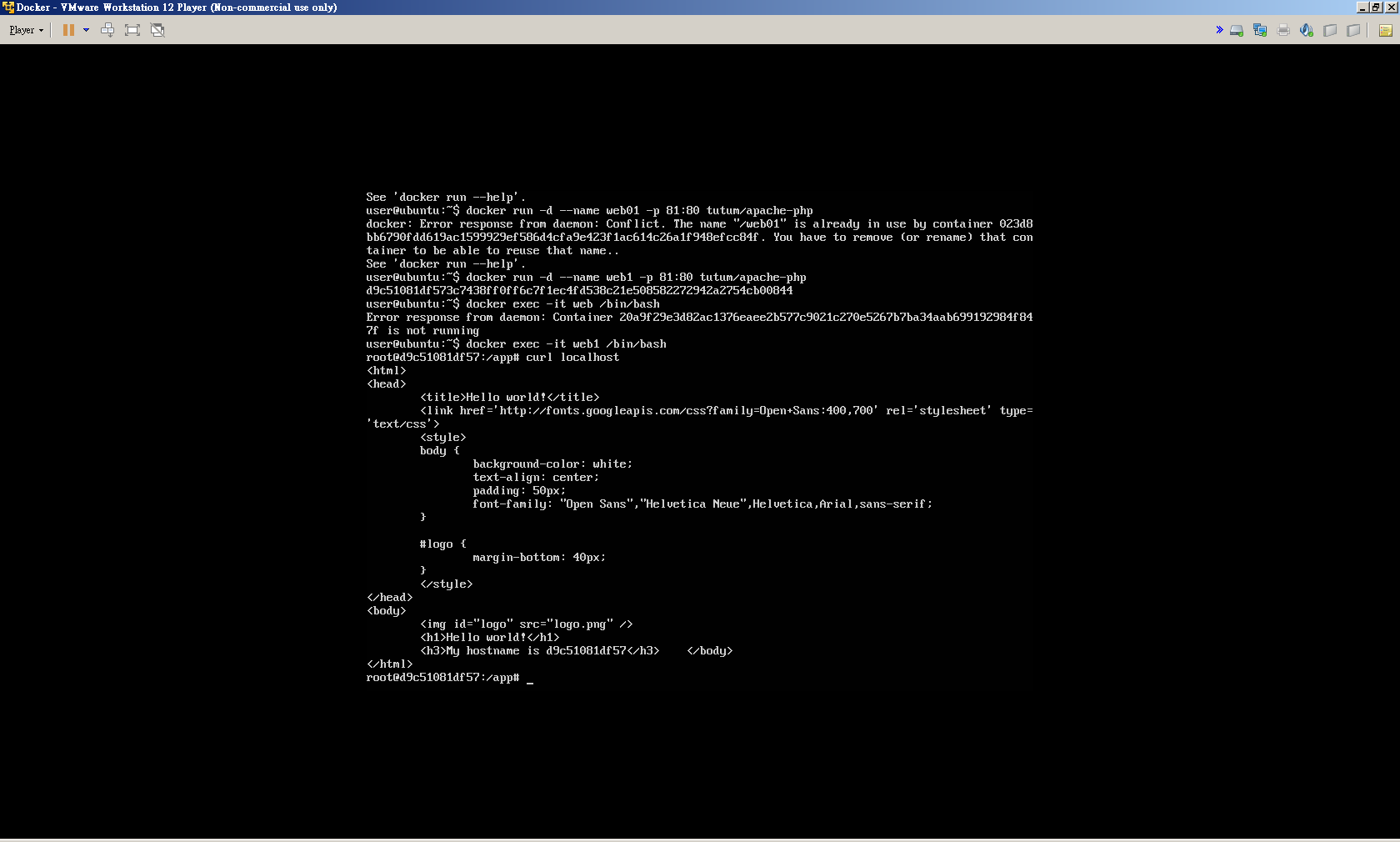
docker run –p <主機阜>: <container阜>容器外部port跟內部port對應

1.docker run –d --name web –p 81:80 tutum/apache-php

(可直接由外部連接到container的第三種方式，建議)

C:\Users\Administrator\AppData\Local\LINE\Cache\tmp\1483006456033.jpg

2.主機存取Container的80埠 (curl 172.17.0.2:80)





docker run –p 192.168.1.1:80:80 (只對應這個ip的80至Container80)

docker run –p 192.168.1.1::80(這個ip隨機選一個埠至Container80)

docker run –p 80:80/udp(只映射udp通訊協定)

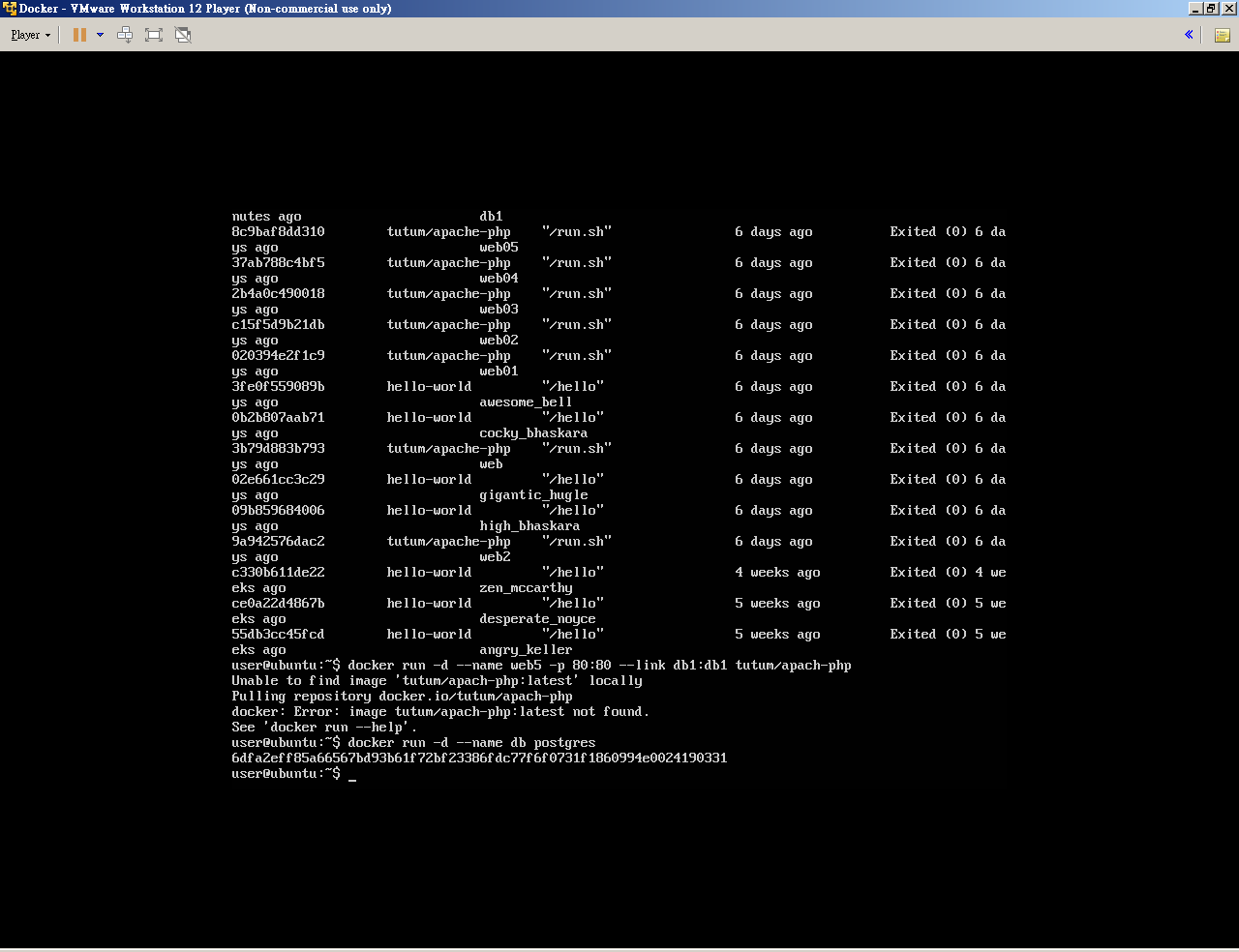
docker run –p 80:80 –p 443:443(多個對應)

docker run –p 192.168.1.1:80:80 –p 10.1.1.1:443:443 (多ip多阜對應)

docker run –p 127.0.0.1:81:80

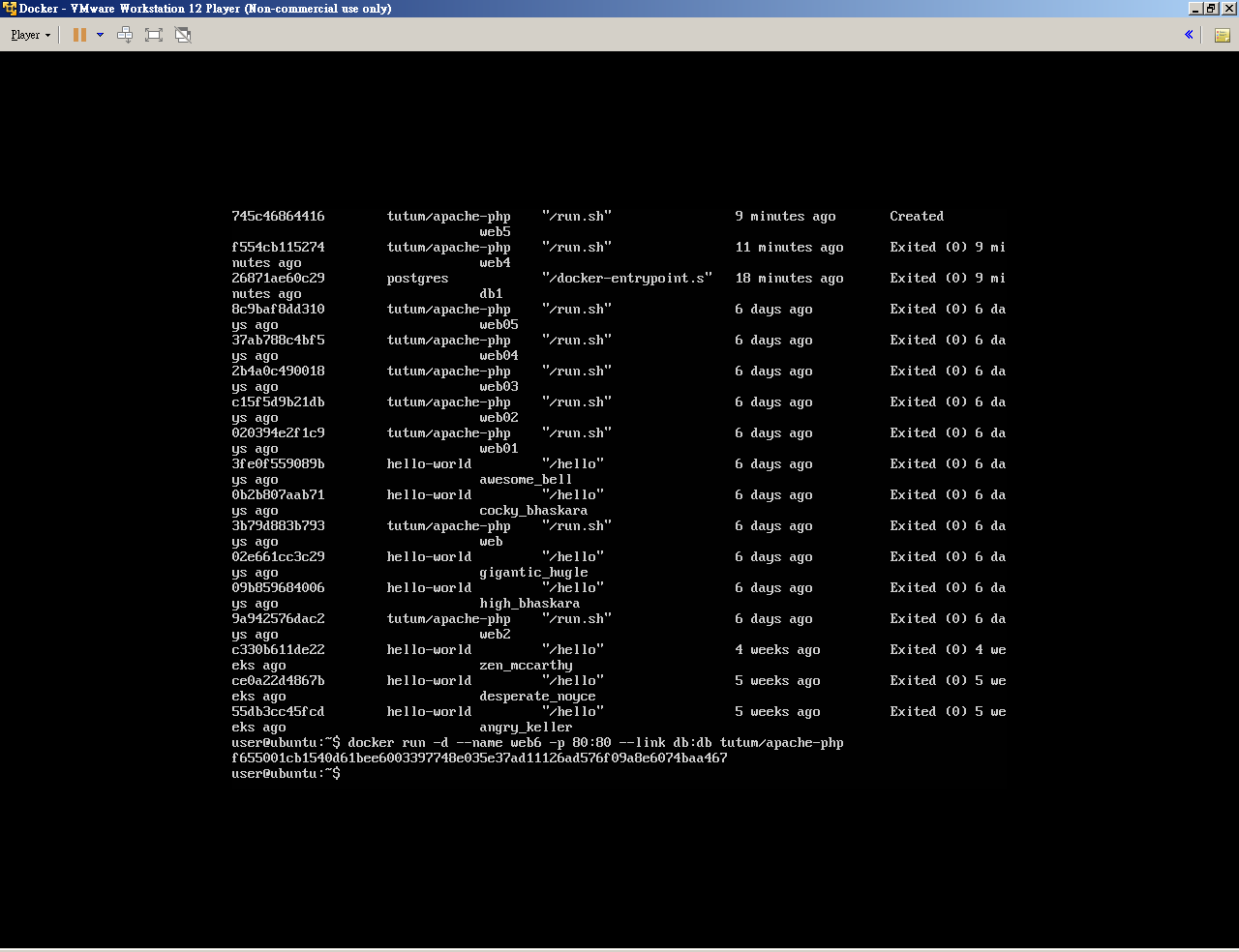
※容器互連

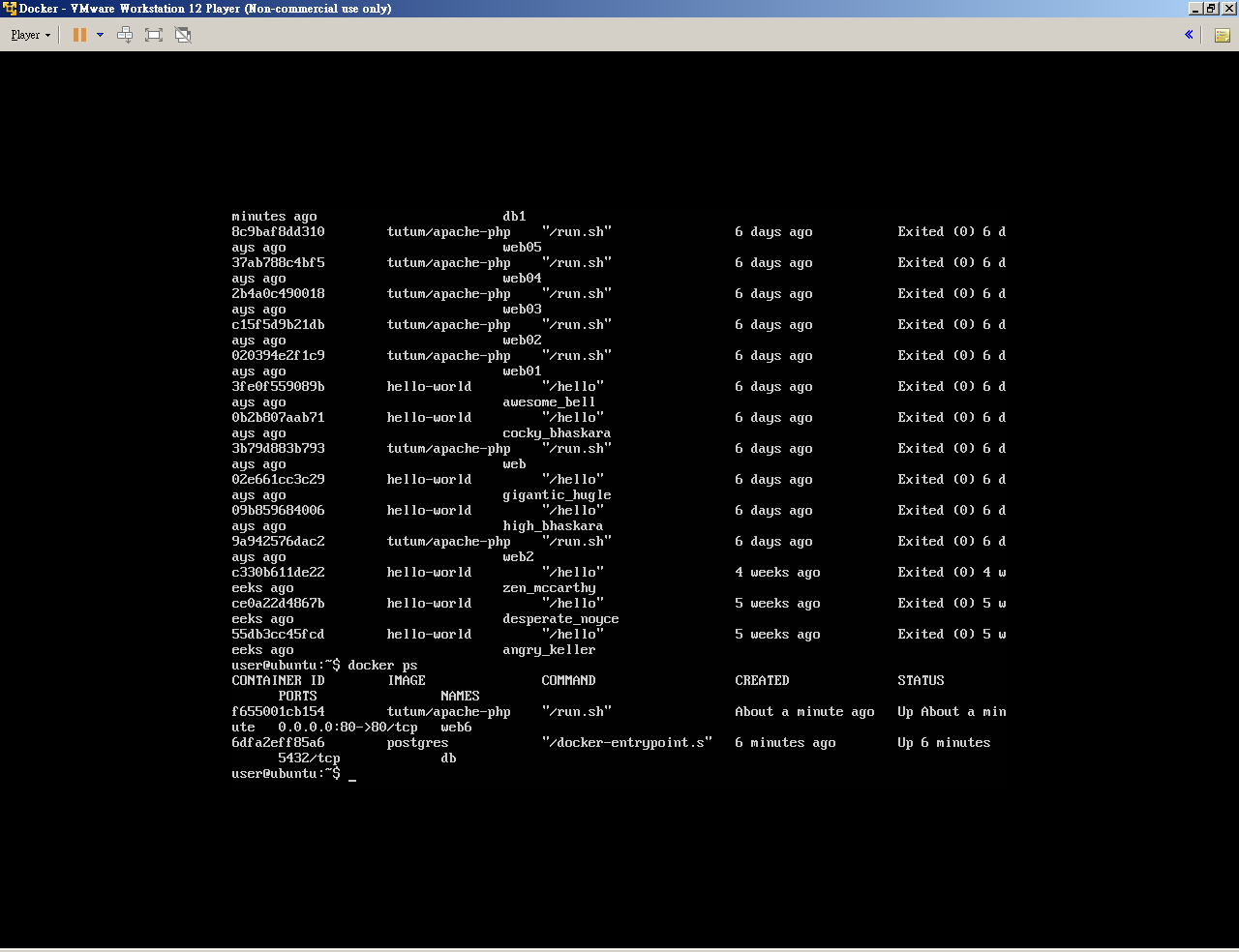
1.docker run –d --name dbpostgres (建立一個資料庫的Container)



2.建立一個web的Container連到db

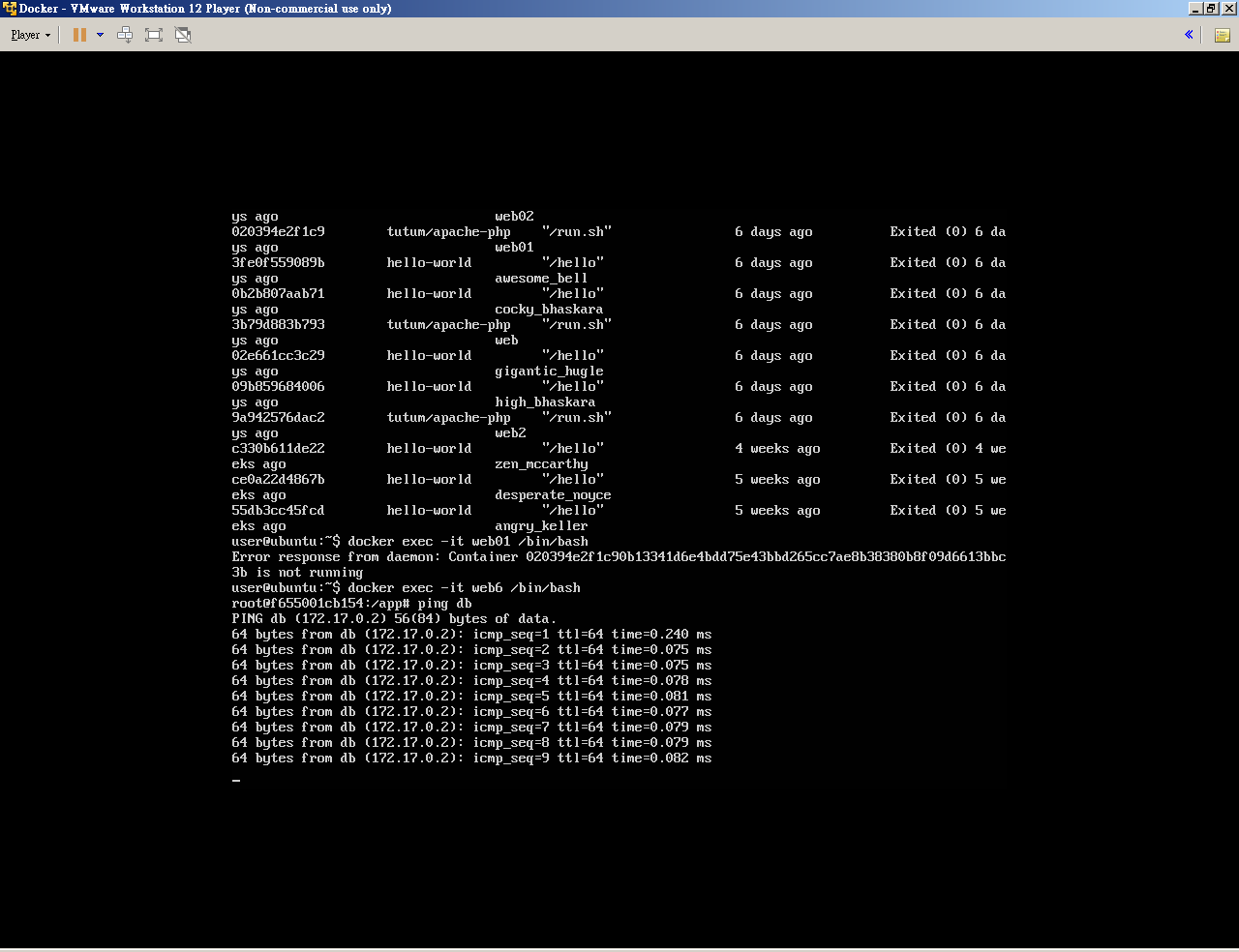
docker run –d --name web –p 80:80 --link db:dbtutum/apache-php



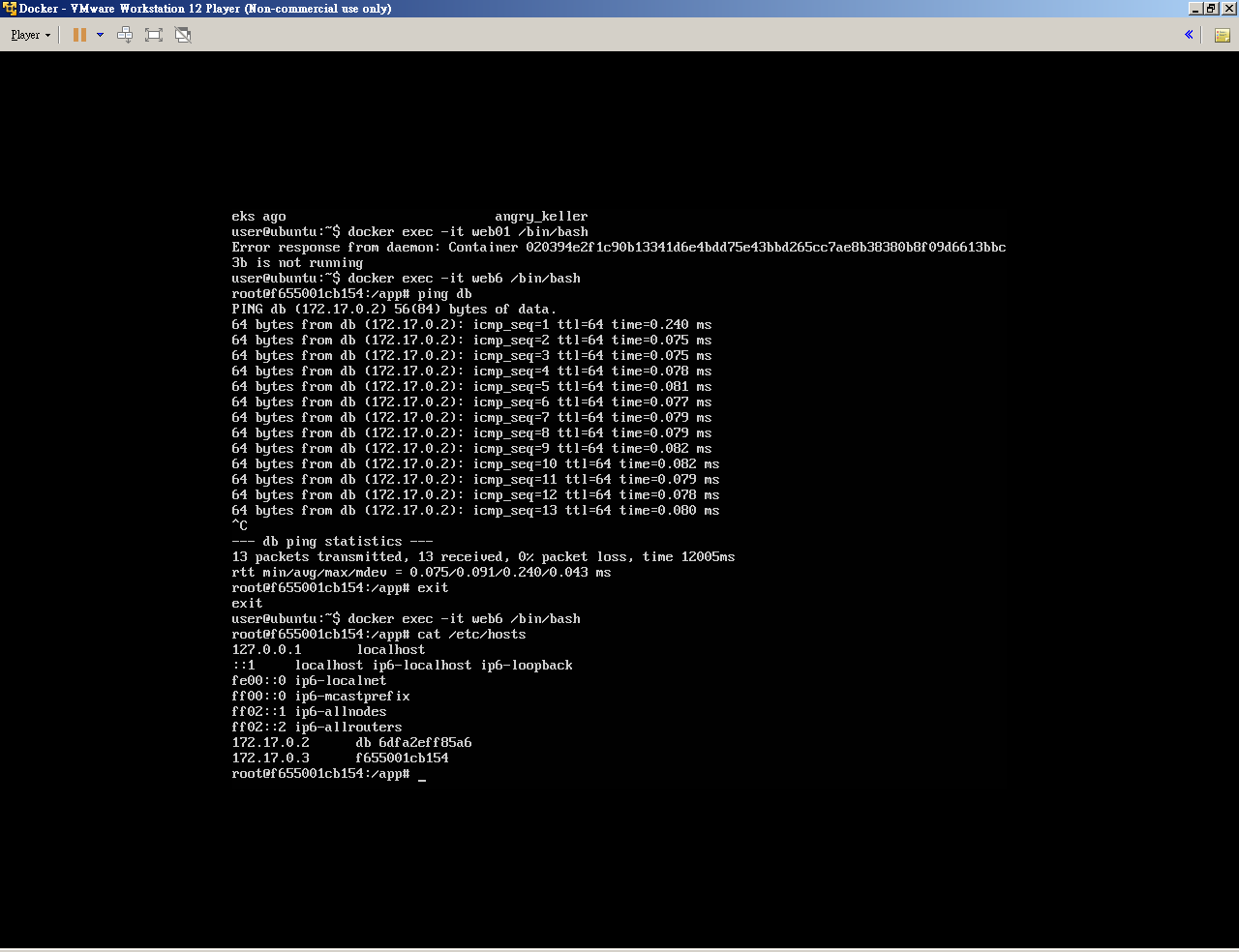


3.可進入web來ping db

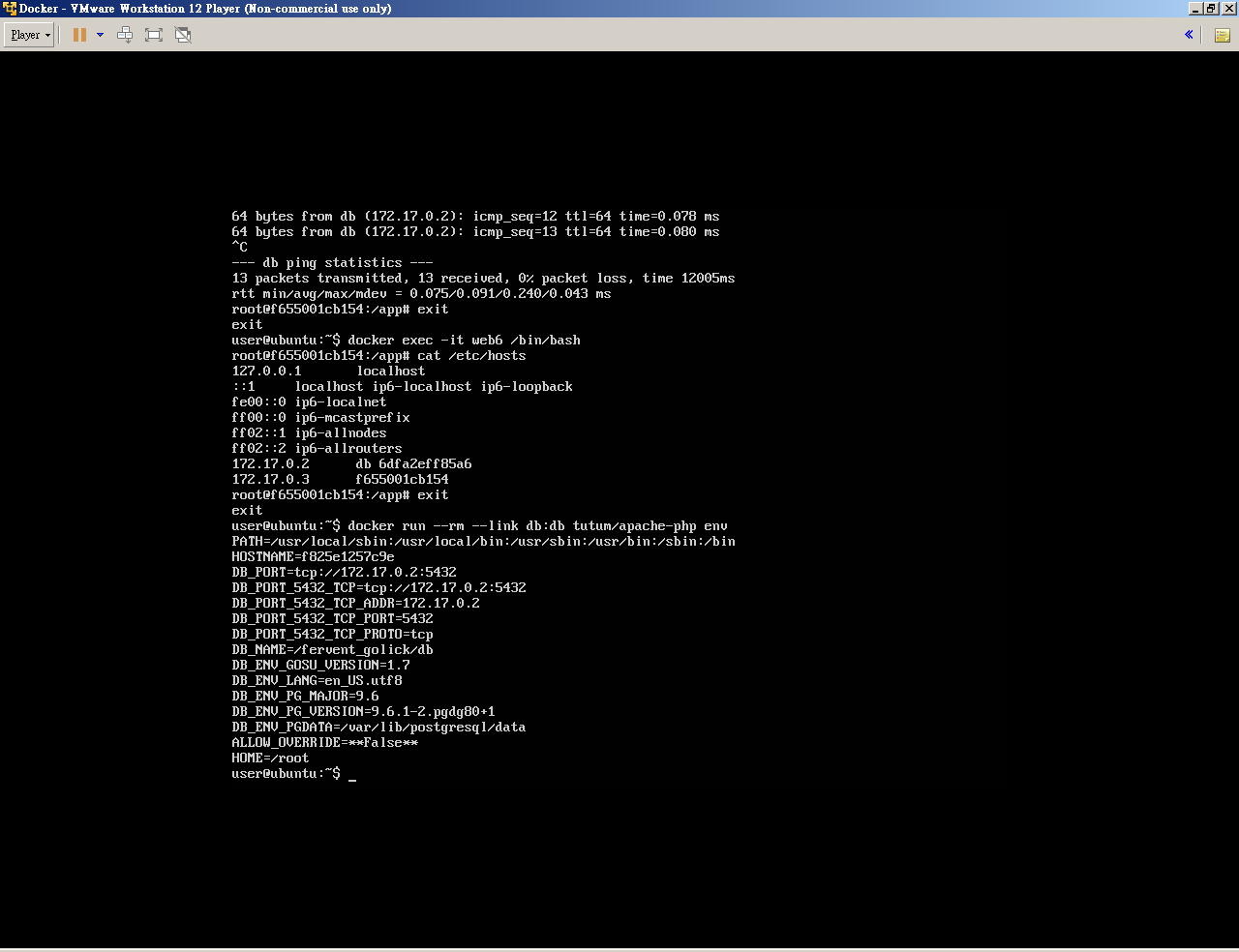
cat /etc/hosts



4.查看子容器/etc/hosts



5.建立一個執行完即消失的Container查看環境變數



105/12/22

data volumes (資料卷) (容器中的一個目錄)

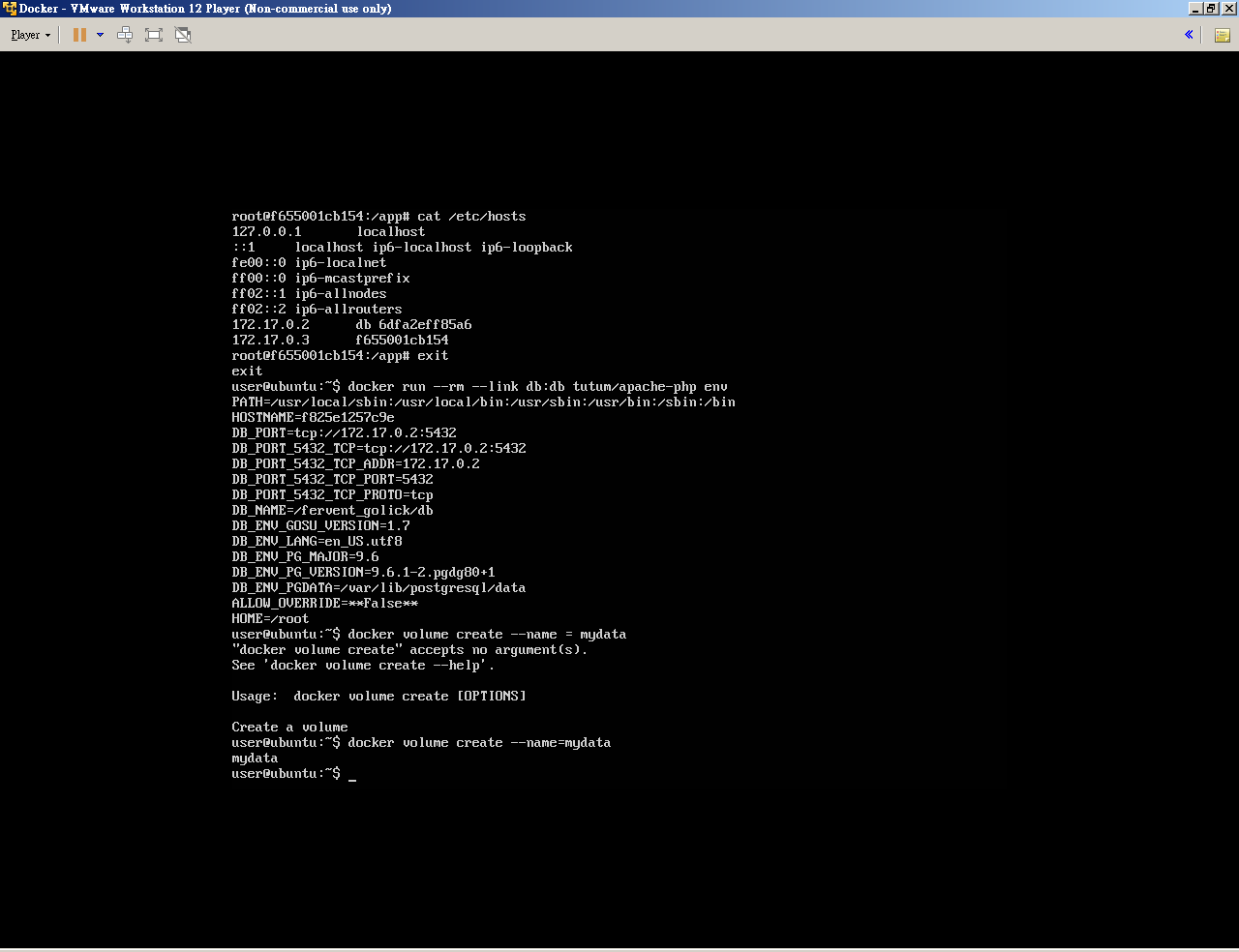
data volume containners(資料卷容器)

三種建立data volume的方式：

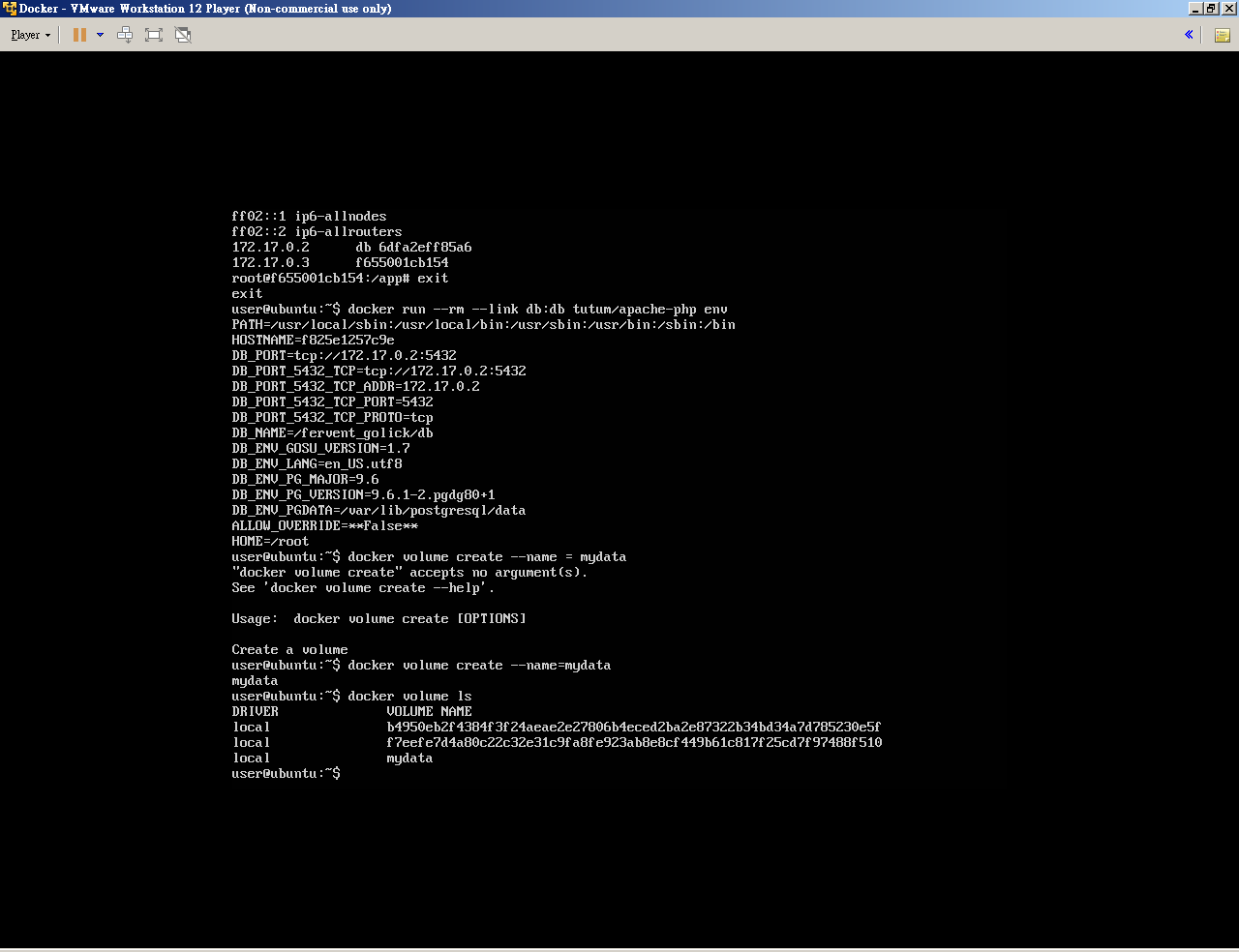
(1) docker volume create

(2)使用container建立

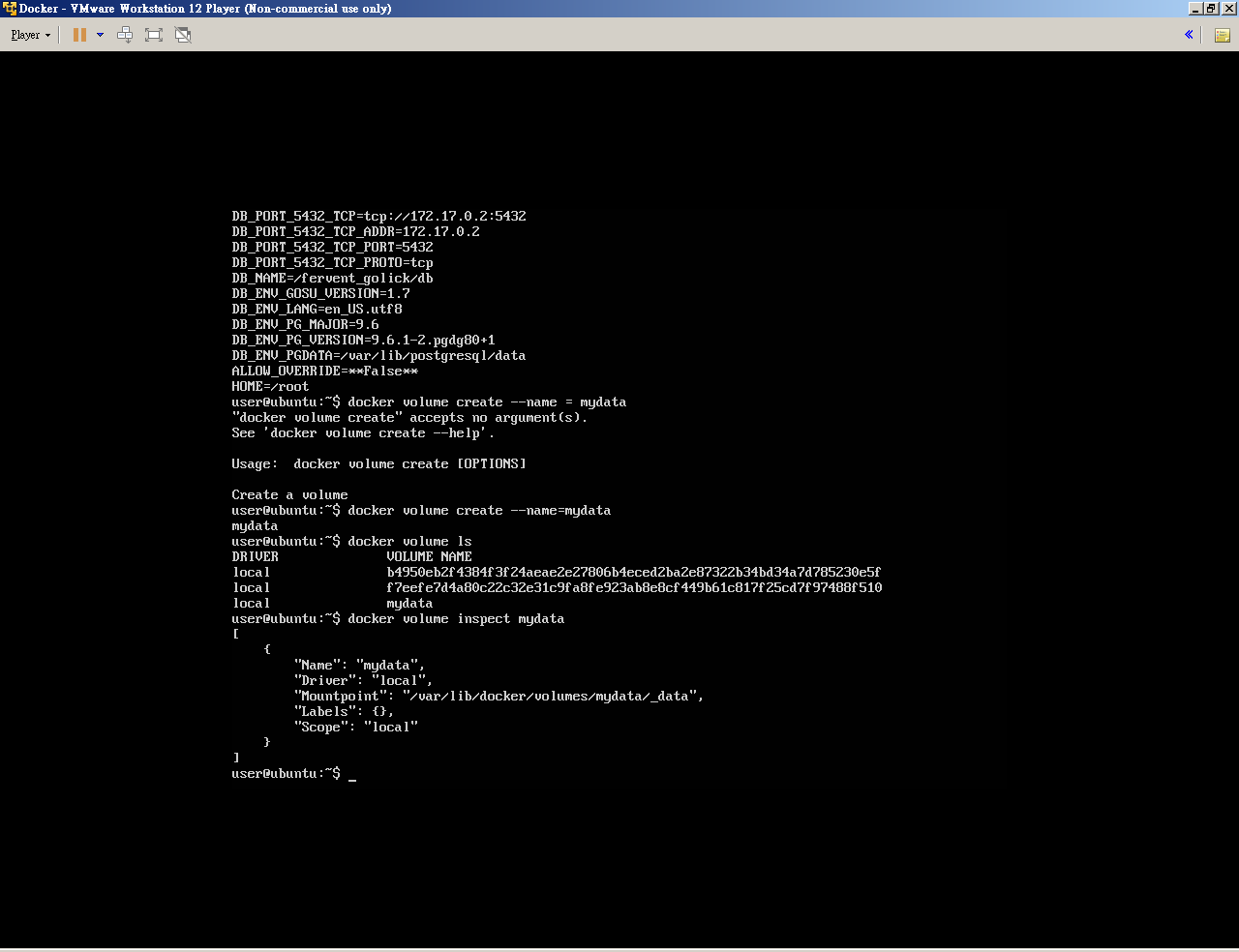
1.docker volume create --name=mydata



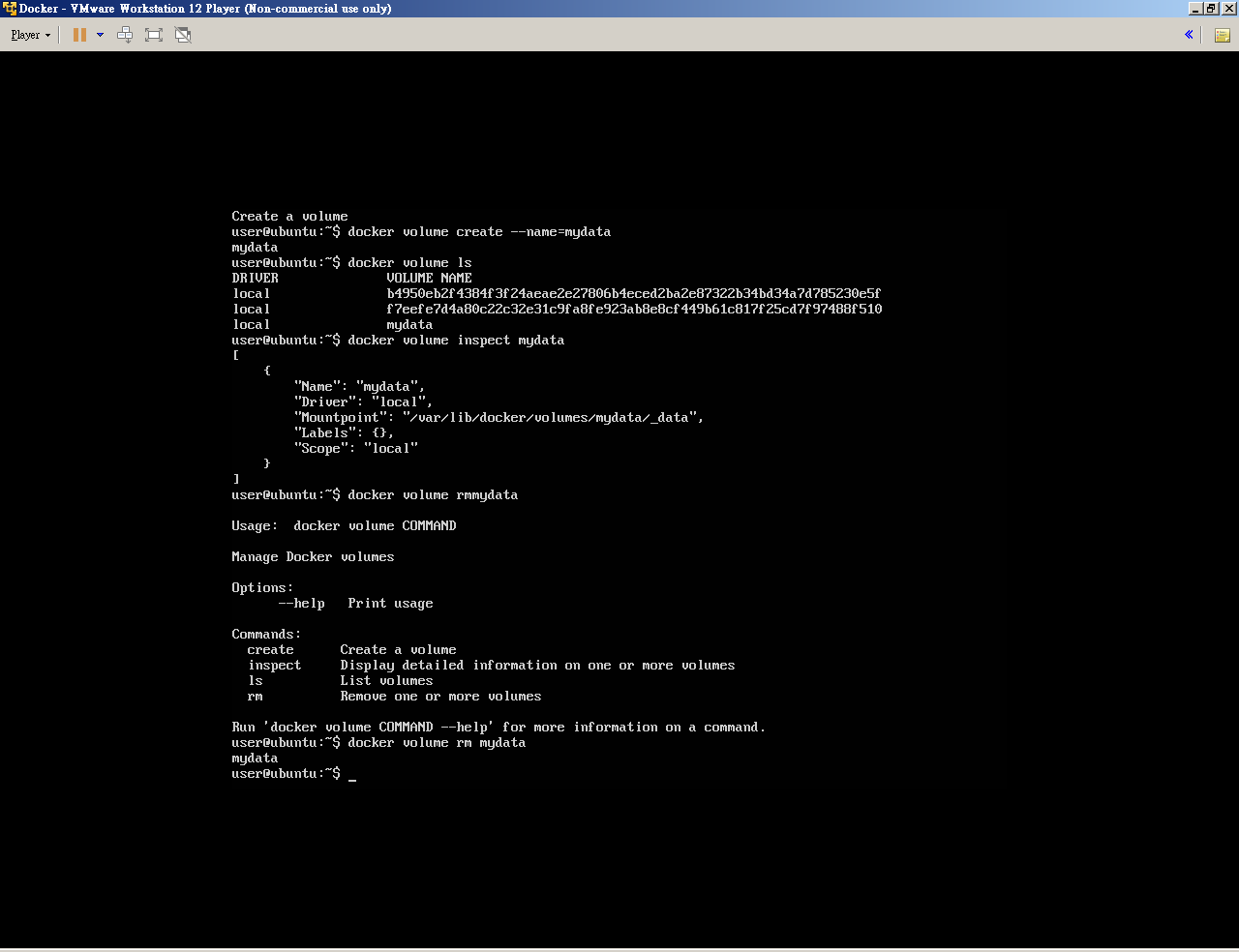
2.docker volume ls (列出所有volume)



3.docker volume inspect mydata (列出mydata的詳細資料)



4.docker volume rmmydata (刪除mydata volume)



5.docker volume rm $(docker volume ls –q) (刪除所有的data volumes)



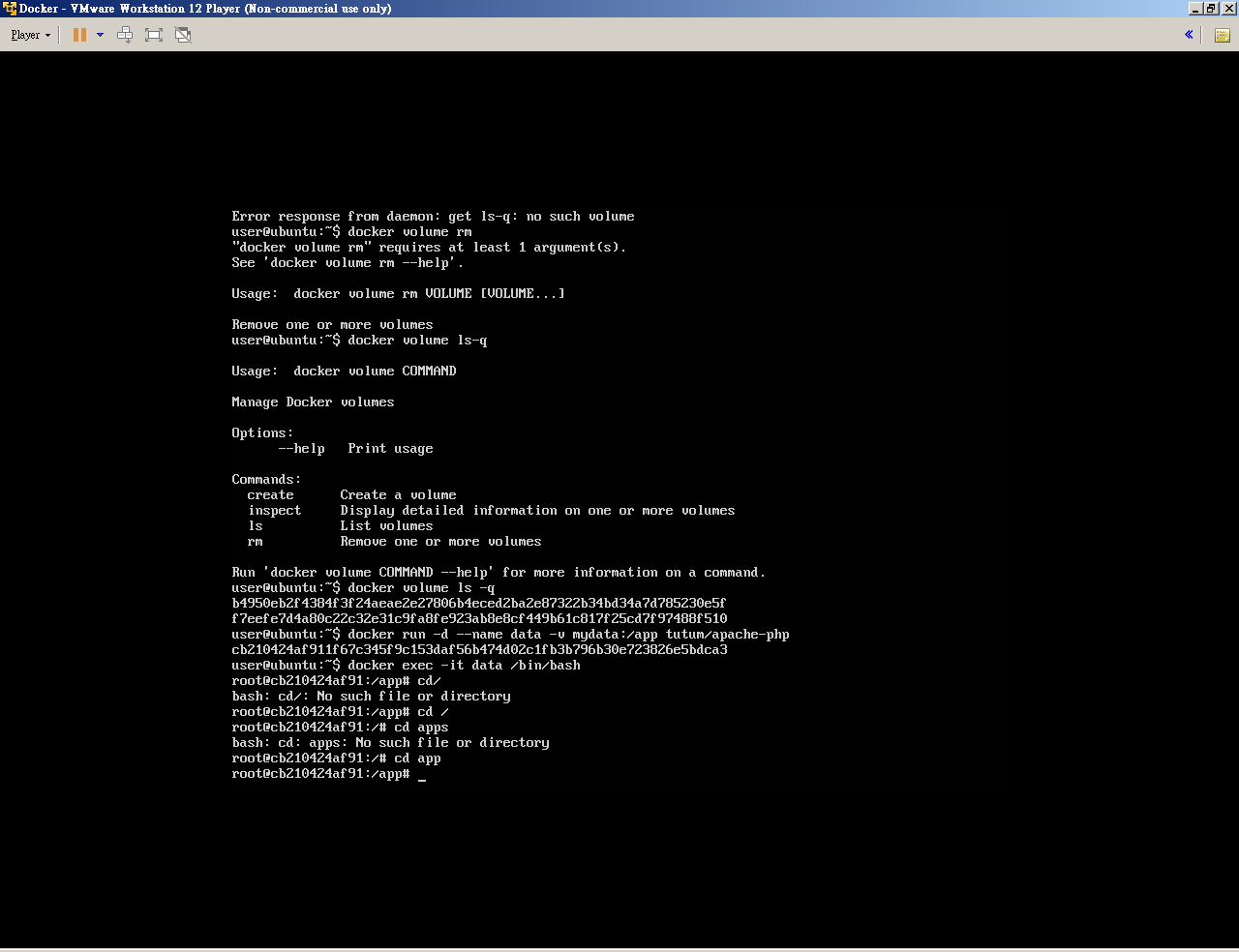
命令一：docker run –d --name data –v mydata:/apptutum/apache-php

(用mydata取代tutum apache裡面的/app目錄)

docker exec –it data /bin/bash

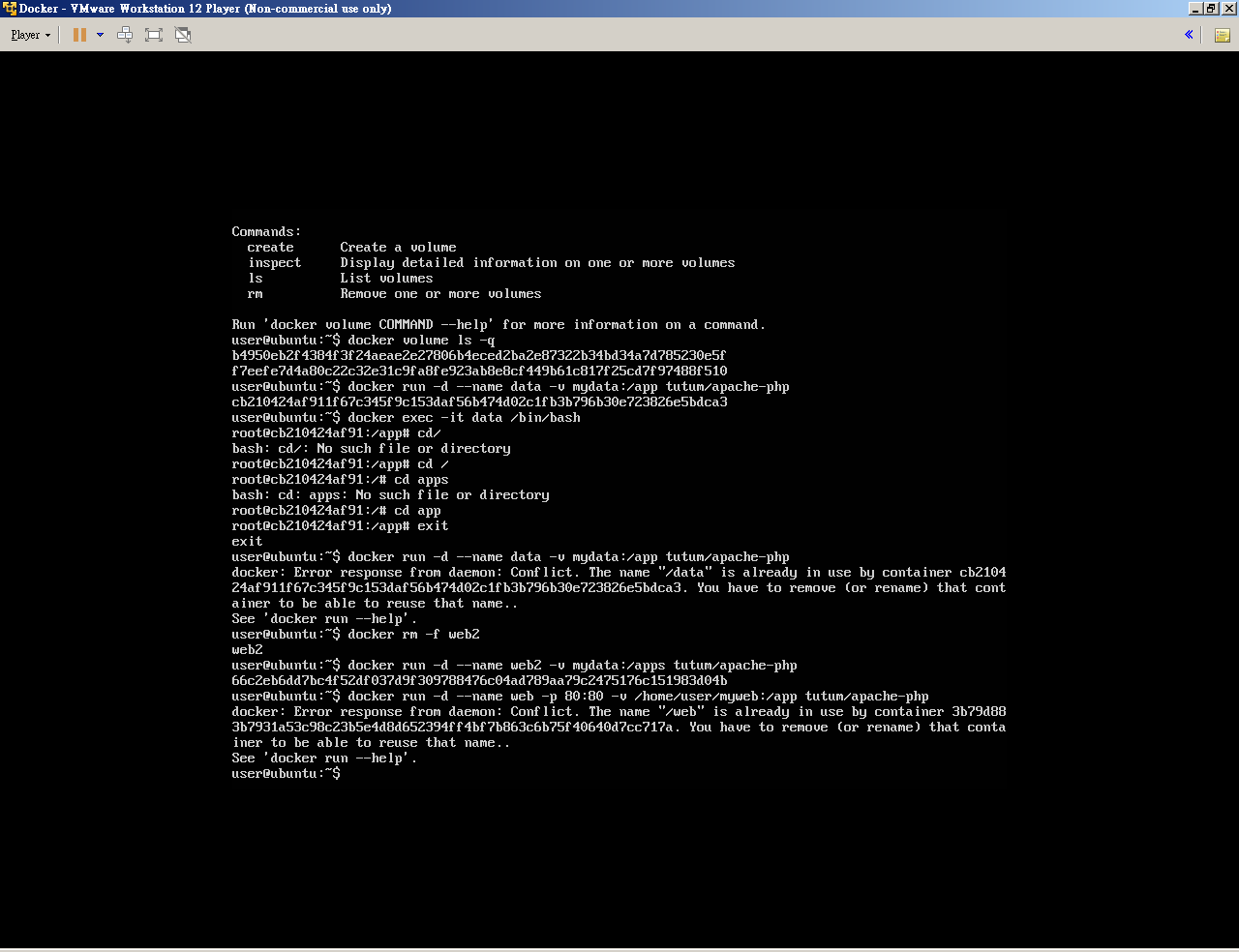
cd/

cd apps



命令二：docker run –d --name web2 –v mydata:/apps tutum/apache-php

(前一個命令一已經新增過mydata，這邊的mydata會直接使用)

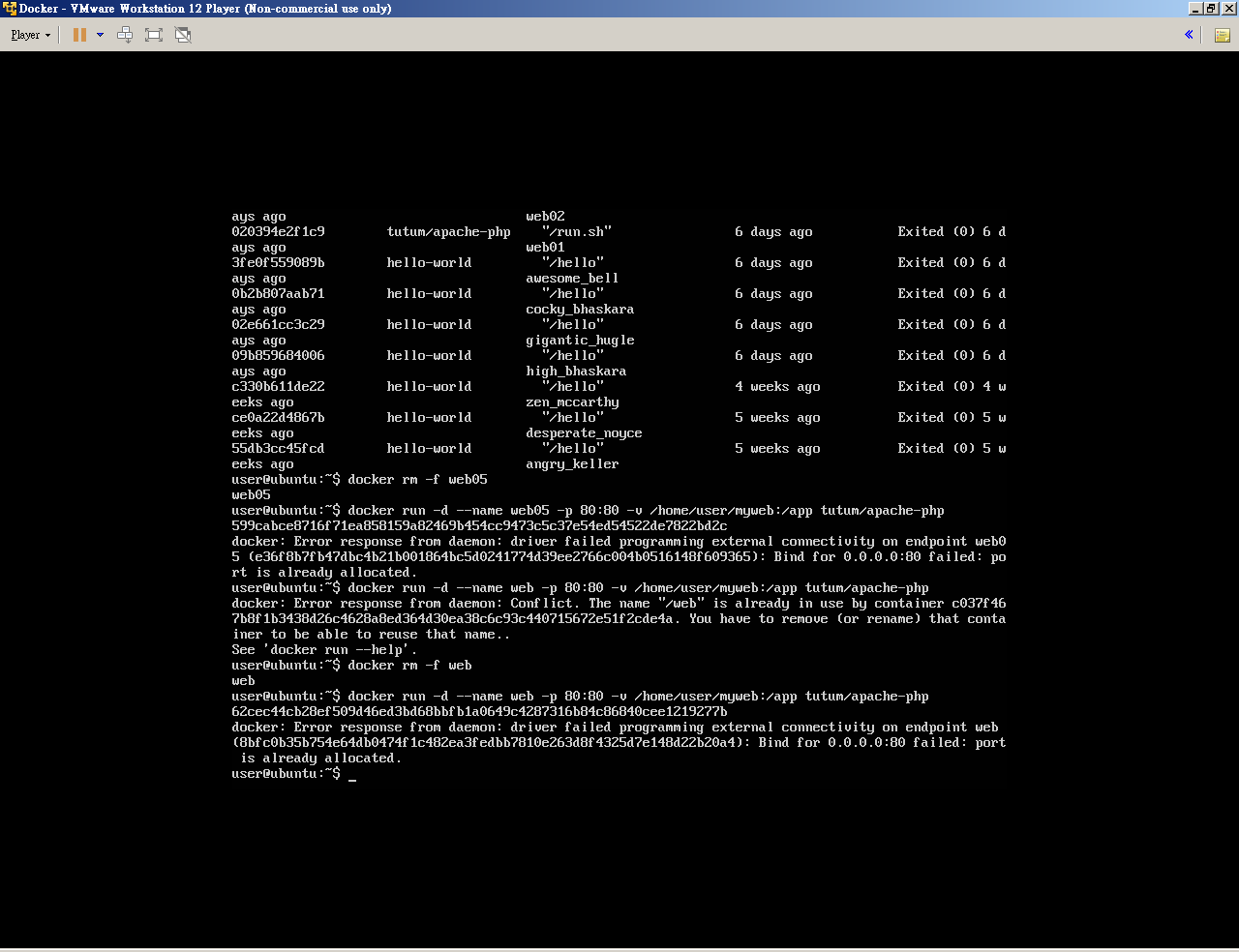


Docker run –d --name web –p 80:80 –v /home/user/myweb:/app tutum/apache-php

-v <主系統路徑>:<容器內路徑>

-v <data volume名稱>:<容器內路徑>

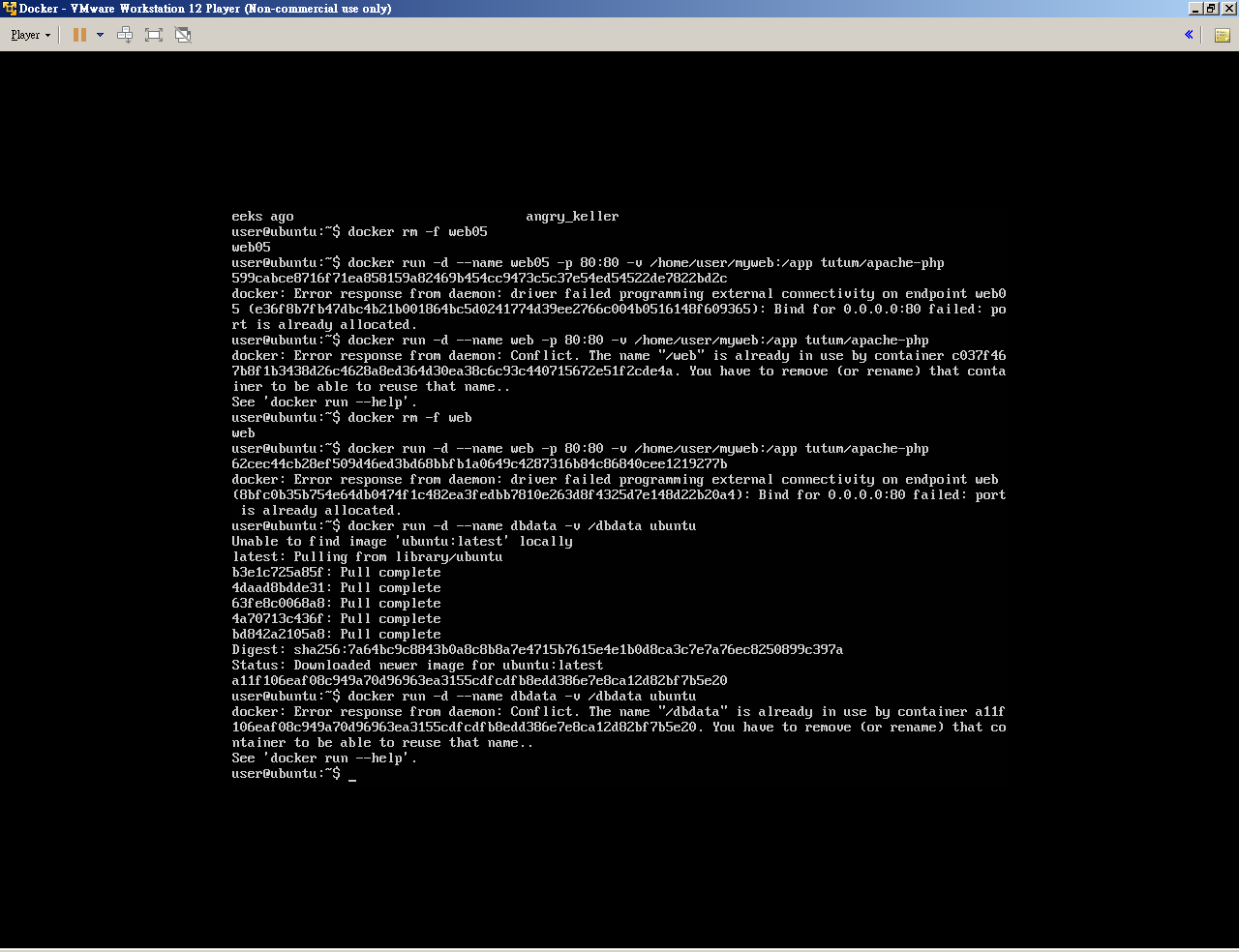
可以在容器目錄後加上:ro (唯獨) :rw (唯寫)



Data volume + container = 資料卷容器

1.docker run –d --name dbdata –v /dbdataubuntu

(用一個ubuntu來掛載dbdata，此為資料卷容器)



2.docker run –d --name db1 --volumes-from dbdatamysql

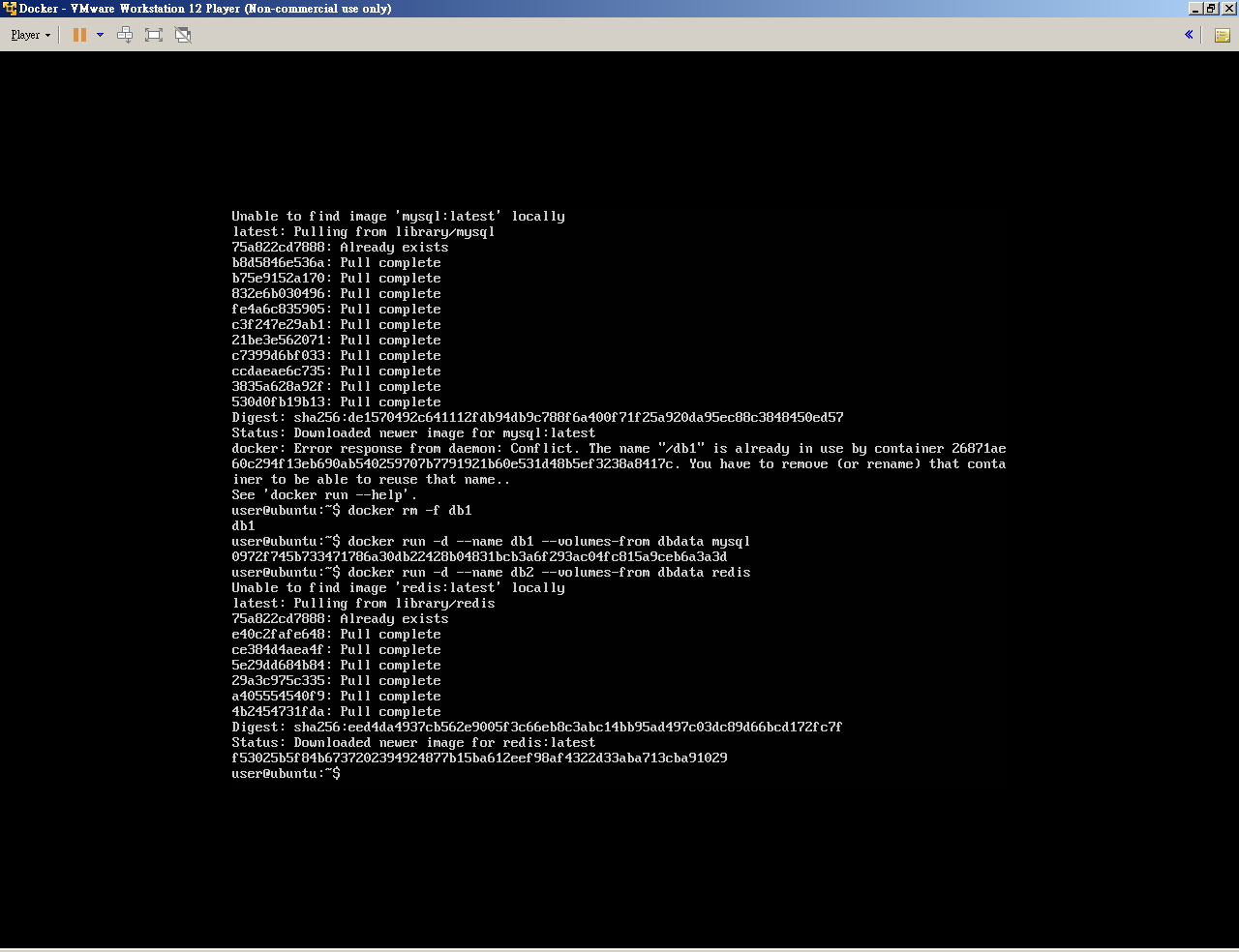
(當其他容器要使用,則用2.3)



3.docker run –d --name db2 --volumes-from dbdataredis

(-volumes-from 把mysql跟redis的儲存位置設為dbdata)

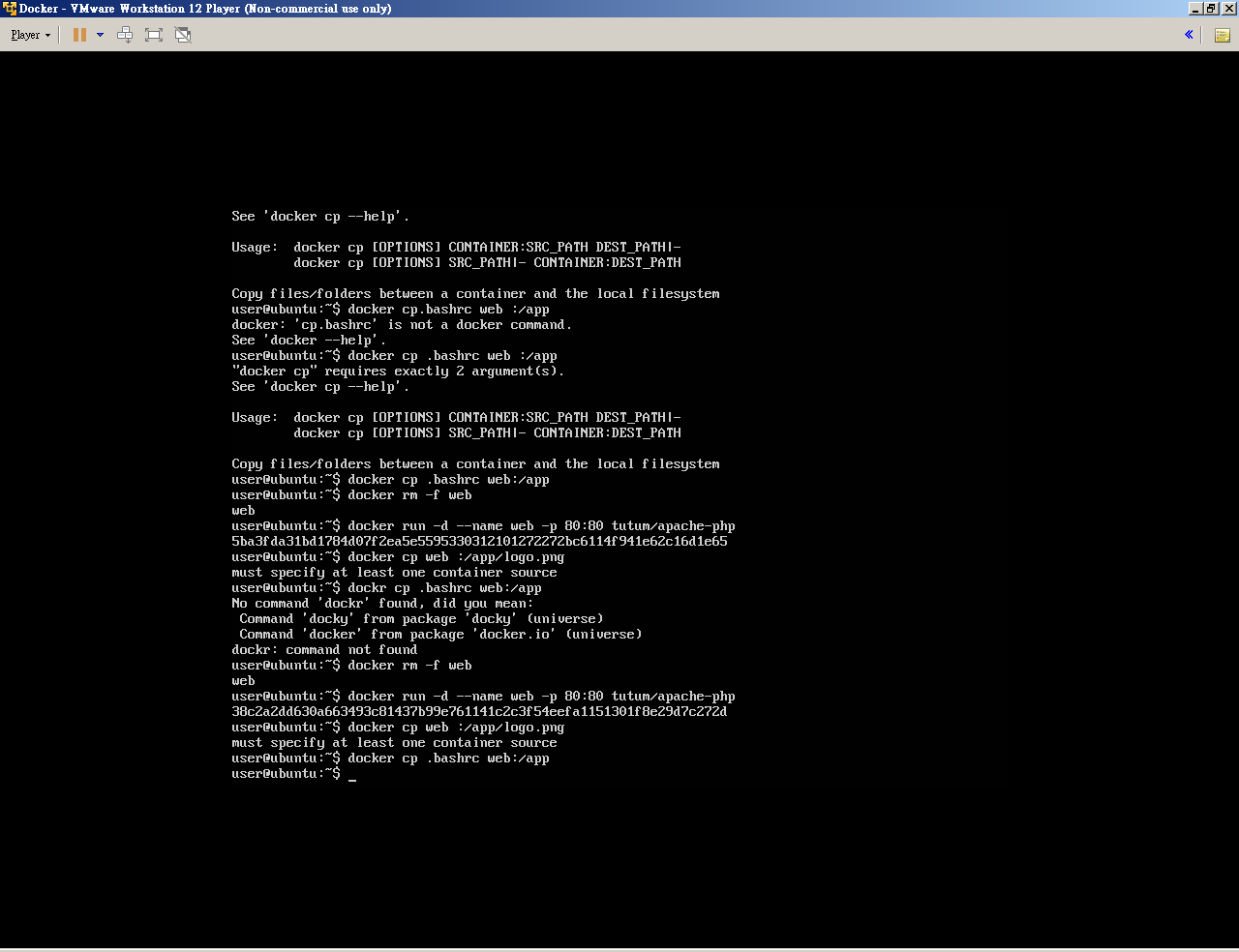
(好處:當dbdata壞掉時，db2跟db1只要掛載dbdata即可，不需要管絕對路徑)



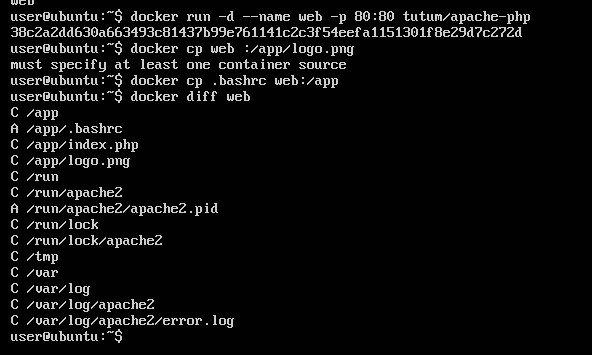
＊具延伸功能，掛載了資料卷的容器db1/db2也可當資料卷

4.docker attach <容器名稱> (連接到執行中的容器)

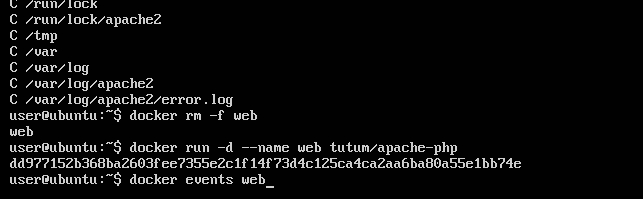
5.docker cp<容器名稱> (在主系統和容器之間複製資料)

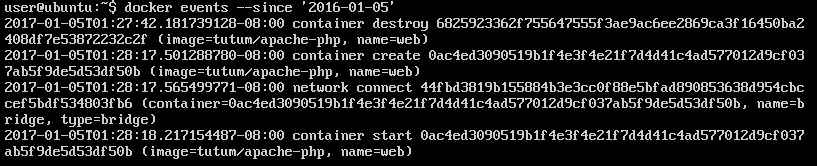


6.dockr diff <容器名稱> (查看主系統和容器的檔案差別)



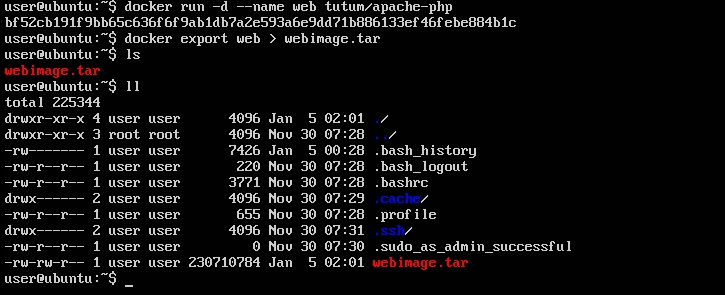
1. docker events <容器名稱> (查看容器的系統狀態)





1. docker export <容器名稱><file.tar> (將容器的檔案系統輸出到壓縮檔)

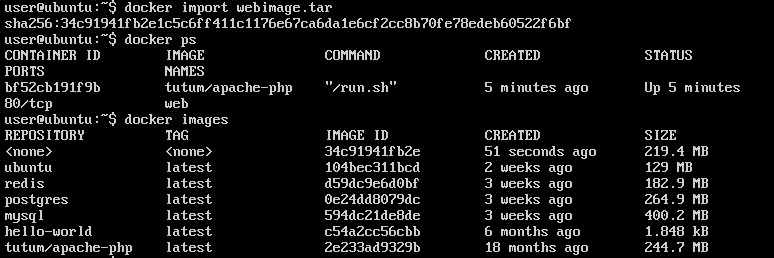
docker export web > webimage.tar



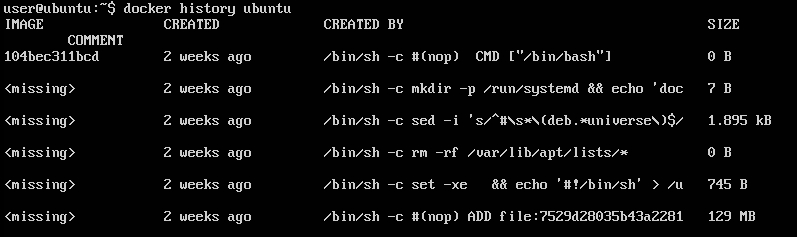
1. docker import <file.tar> (從壓縮檔還原成映像檔)

docker import webimage.tar

docker images



1. docker history <image file> (列出一個映像檔的修改歷史)



11.docker images (列出映像檔)



只列出映像檔id:



1. dockerrmi<映像檔> (刪除映像檔)

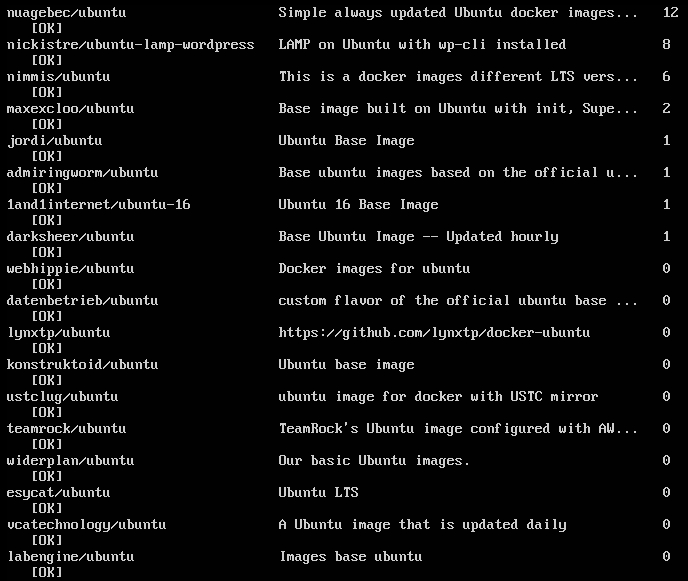


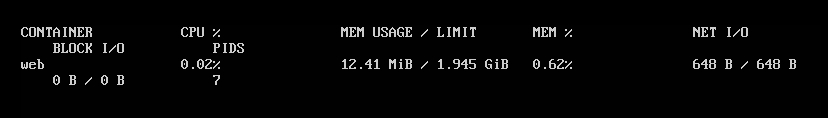
1. docker kill <容器名稱> (中止執行中容器)
2. docker port <容器名稱>



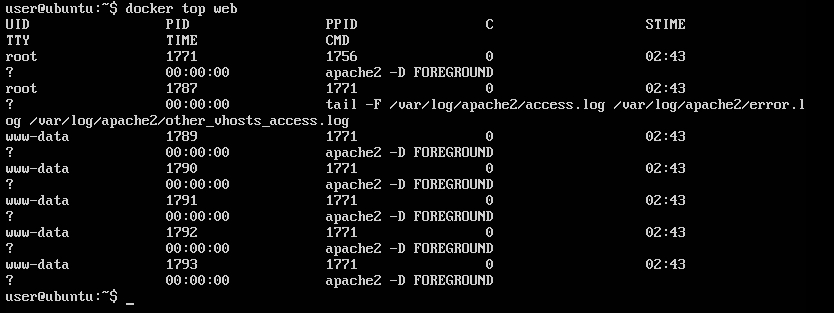
1. docker search <關鍵字> (在Docker Hub上尋找映像檔)

C:\Users\Administrator\AppData\Local\LINE\Cache\tmp\1483613223379.jpg



15.docker stats <container> (觀看容器的系統資源)

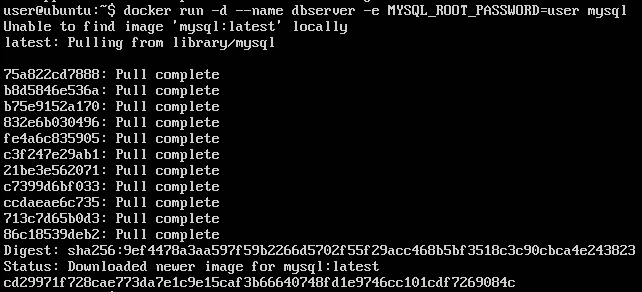
16.docker top <container> (查看容器內部資源使用)



105/12/29

1.docker run –d --name dbserver –e MYSQL\_ROOT\_PASSWORD=user mysql

(用環境變數設定mysql密碼為user)



2.docker exec –it dbserver bash (進入mysql)  
C:\Users\吳青宗\AppData\Local\LINE\Cache\tmp\1484047404061.jpg

3.mysql–uroot–p (用mysql client連入mysql)



4.docker run -d --name web --link dbserver: mysql tutum/apache-php

(dbserver設好的db container，mysql自己取的名稱)

C:\Users\吳青宗\AppData\Local\LINE\Cache\tmp\1484049845142.jpg

5.docker exec –it web bash

6.ping mysql (ping的到代表可由web連到mysql)

7.docker run –d –name client—link bserver:mysql –e (不用換行,接下去打第二行) MYSQL\_ROOT\_PASSWORD=user mysql

C:\Users\吳青宗\AppData\Local\LINE\Cache\tmp\1484050849348.jpg

9.docker exec –it client bash

10.mysql–uroot–p –h mysql (連入mysql)

11.docker exec –it client mysql–uroot–p –h dbserver (可用docker exec直接run mysql命令)

12.docker exec –it dbserver cat /var/log/mysqll/error.log (範例：列出log檔案)

touchshhuang.cnf

13.docker run –d --name db2 –e MYSQL\_ROOT\_PASSWORD=user &&

–v shhuang.cnf:/etc/mysql/conf.d/shhuang.cnfmysql

(把shhuang.cnf對應到db2裡面的/etc/mysql/conf.d/shhuang.cnf)

\*\*\*\*\*自建映像檔\*\*\*\*\*\*

第一種方法(只能封頂，少用)

1.docker run –d --name imagetest Ubuntu

C:\Users\吳青宗\AppData\Local\LINE\Cache\tmp\1484053928916.jpg

2.docker exec –it imagetest bash

3.apt-get update

4.apt-get upgrade

(exit離開)

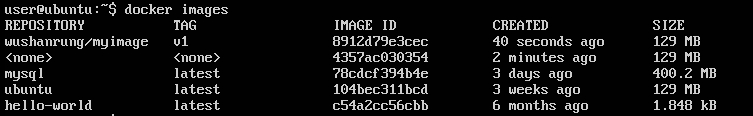
5.docker commit –m “update system” –a “shhuang” imagetest (-a作者名稱，-m是container說明)

C:\Users\吳青宗\AppData\Local\LINE\Cache\tmp\1484055094586.jpg

6.docker commit –m “create abcde” –a “shhuang” imagetestshhuang/myimage:v1

C:\Users\吳青宗\AppData\Local\LINE\Cache\tmp\1484055202918.jpg

7.docker images



8.docker run –d --name test shhuang/myimage:v1

C:\Users\吳青宗\AppData\Local\LINE\Cache\tmp\1484055479774.jpg

第二種方法Dockerfile

#註解

FROM(基底印象檔資訊):從哪個Image建立

MAINTAINER:建立者

RUN:執行程式

EXPOSE:開放通訊阜

ADD:將主機資料加入容器

ENV:傳遞參數到外面的世界，可由docker inspect取得

CMD:容器啟動時指令，通常為.sh指令檔

docker build –t=”映像檔標籤” . ( .代表目前目錄下的docker file)

docker build –t=”shhuang/buildtest:v1” .

docker file example:

FROM ubuntu:trusty (ubuntu每一個版本都有名稱，trusty是14.04版)

MAINTAINER shhuang<shhuang@must.edu.tw>

EXPOSE 80

RUN apt-get update –y

RUN apt-get upgrade -y

EX: docker build –t “shhuang/utest:v1” .

※建立Ubuntu 12的LAMP

1.建立一個目錄來處理這個映像檔

C:\Users\吳青宗\AppData\Local\LINE\Cache\tmp\1484056134472.jpg

2.建立資料目錄以便放入映像檔

C:\Users\吳青宗\AppData\Local\LINE\Cache\tmp\1484056081459.jpg

3.在資料目錄中放入測試用的網頁

C:\Users\吳青宗\AppData\Local\LINE\Cache\tmp\1484056861932.jpg

內容如下(此檔案為測試php版本用)





編輯模式下，按下Ctrl-O存檔，Ctrl-X離開

4.容器啟動內部自動執行檔

回到ubuntu12lamp目錄下，輸入nano run.shC:\Users\吳青宗\AppData\Local\LINE\Cache\tmp\1484057369938.jpg

輸入指令: #!/bin/bash

chown www-data:www-data /app -R

**(將容器/app下的使用者權限改成www-data)**

source /etc/apache2/envvars

**(設定apache2的啟動環境變數)**

exec apache2 -D FOREGROUND

**(在容器中啟動apache)**



編輯模式下，按下Ctrl-O存檔，Ctrl-X離開

5.回到ubuntu12lamp目錄下，輸入 nano Dockerfile

C:\Users\吳青宗\AppData\Local\LINE\Cache\tmp\1484057835649.jpg

輸入以下指令:

# This dockerfile uses the ubuntu 12.04image

# VERSION 1 - EDITION 1

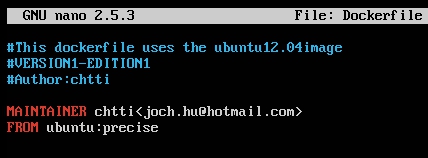
# Author: chtti

#維護者

MAINTAINER chtti <josh.hu@hotmail.com>

#映像檔來源ubuntu 12.04，官方代號為precise

FROM ubuntu:precise



#安裝基本環境

ENV DEBIAN\_FRONTEND noninteractive **(設定環境變數)**

RUN apt-get update && \

apt-get -yq install \

curl \

apache2 \

libapache2-mod-php5 \

php5-mysql \

php5-gd \ **(安裝基本的LAMP套件)**

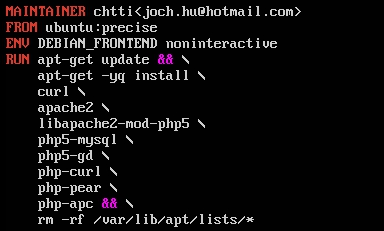
php5-curl \

php-pear \

php-apc && \

rm -rf /var/lib/apt/lists/\*  **(清乾淨更新完後的垃圾)**

**(注意：此時這個RUN指令完成後，就會生成一個layer)**



RUN sed -i "s/variables\_order.\*/variables\_order = \"EGPCS\"/g" /etc/php5/apache2/php.ini

RUN curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --filename=composer

以上指令為設定php.ini中的數值，以及下載php，並且設定安裝執行時的目錄

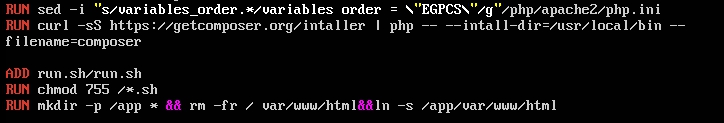
接下將容器啟動時的的執行檔run.sh加入映像檔的根目錄中

ADD run.sh /run.sh

變更該sh的檔案權限

RUN chmod 755 /\*.sh

刪掉預設目錄，建立目錄以及指向目錄的連結

RUN mkdir -p /app && rm -fr /var/www/html && ln -s /app /var/www/html  


ADD sample/ /app (將主機/home/chtti/ubuntu12lamp/sample的目錄加到容器/app目錄下)

EXPOSE 80 (開放容器的80port)

WORKDIR /app (設定容器的工作目錄為/app)

CMD ["/run.sh"] (容器啟動時執行程式)

