

Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores

(Computer Engineering Academic Area)

Programa de Licenciatura en Ingeniería en Computadores

(Licentiate Degree Program in Computer Engineering)

Curso: CE-4303 Principios de Sistemas Operativos

(Course: CE-4303 Principles of Operative Systems)



## Manual de Uso Tarea Corta 2: Container Jobs

(User manual Homework 2: Container Jobs)

Realizado por:

Made by:

Giovanni Villalobos Quirós 2013030976

Profesor:

(Professor)

Ing. Alejandra Bolaños Murillo

Fecha: Cartago, Octubre 24, 2017

(Date: Octubre 24th, 2017 )

# Introducción.

El proyecto fue desarrollado en todos los requerimientos especificados. Algunos aspectos importantes a tomar en cuenta.

- El programa contiene en la carpeta Cliente ciertas imágenes que fueron los test realizados, en caso de querer probar estas imágenes simplemente es necesario ingresar su nombre con extensión, cuando se solicite la ruta, siempre y cuando se corra el cliente dentro de dicha carpeta, de otra manera es necesario introducir la ruta completa.
- Las imágenes probadas van del rango de los 150x150 pixeles hasta los 3000x3000 pixeles, logrando los resultados esperados en todas las pruebas.
- Las imágenes a enviar pueden ser de cualquier extensión
- En el archivo de configuración debe estar en la dirección del volumen que se va a montar puesto que antes de correr el server se debe colocar el archivo de configuración que se encuentra en el comprimido en la dirección que se usará para dicho volumen.
- En el archivo de configuración no se debe cambiar el orden de las oraciones que dicen "Accepted:" y "Not Trusted:" simplemente se deben cambiar las direcciones que aparecen en dicho archivo o agregar más de la misma manera en que aparecen ahí separadas por un cambio de línea.

## 1. Cargar imagen del contenedor

En la carpeta comprimida se encuentra el archivo comprimido llamado *"img\_server\_classifier.tar"*, el cual es la imagen que los contenedores deben correr al ejecutarse. Por lo tanto es necesario cargar dicha imagen a docker con el siguiente comando en la carpeta donde se encuentre el comprimido.

```
sudo docker load -i="img_server_classifier.tar"
```

Dicho comando carga la imagen previamente hecha para poder ejecutar los contenedores con dicha imagen. A continuación se muestra un ejemplo de uso.

```
giovanni@giovanni-CE ~/Desktop $ sudo docker load -i="img_server_classifier.tar"
cf516324493c: Loading layer 205.2MB/205.2MB
f91b5268d800: Loading layer 1.536kB/1.536kB
f341812c36ca: Loading layer 23.04kB/23.04kB
d8f334e58971: Loading layer 58.28MB/58.28MB
d3173b877787: Loading layer 20.81MB/20.81MB
f0c11052b06a: Loading layer 30.18MB/30.18MB
00278f46e232: Loading layer 113.6MB/113.6MB
d6caa8c3dcbe: Loading layer 111.8MB/111.8MB
714b2b7c32e9: Loading layer 24.97MB/24.97MB
Loaded image: img_server_classifier:latest
giovanni@giovanni-CE ~/Desktop $
```

## 2. Ejecución Server

Para ejecutar el programa primero se debe correr el contenedor, luego de haberlo importado se ejecuta en terminal:

```
sudo docker run -it -p 6666:6666 -v /Users/<path>:/carpetaDocker/ img_server_classifier
```

Aquí como vemos se corre un contenedor tomando como base la imagen *img\_server\_classifier* sobre la cual fue construido usando el DockerFile adjunto. Seguidamente el servidor empezará a correr en el puerto 6666 de la computadora donde se ejecute. El output mostrado es el siguiente

```
giovanni@giovanni-CE ~/Documents/git/Image-RGB-Classifer-on-Container
giovanni@giovanni-CE ~/Documents/git/Image-RGB-Classifer-on-Container $ sudo docker run -it -p 6666:6666
-v /Users/img_classifier:/carpetaDocker/ img_server_classifier
Using port: 6666
Server Network info:
Possible Local Address: lo: 127.0.0.1
Possible Local Address: eth0: 172.17.0.2
-----Server Started-----
^[]
```

## 3. Ejecución Cliente

### 3.1 Manera 1: Utilizando python

Para el programa del cliente se debe dirigirse a la carpeta llamada “*Client*” la cual contiene el código python del cliente. Para poder ejecutarlo debe tenerse instalado python en linux para la computadora que lo desee ejecutar. Como verá esta carpeta contiene una serie de imágenes de prueba que fueron usado como test. Se procede a ejecutar el siguiente comando:

```
python client.py <ip del server>
```

Para conseguir la ip del server únicamente se debe ejecutar el comando *ifconfig* en terminal de linux en la máquina que ejecute el server y usar el ip correspondiente.

### 3.2 Manera 2: Utilizando ejecutable

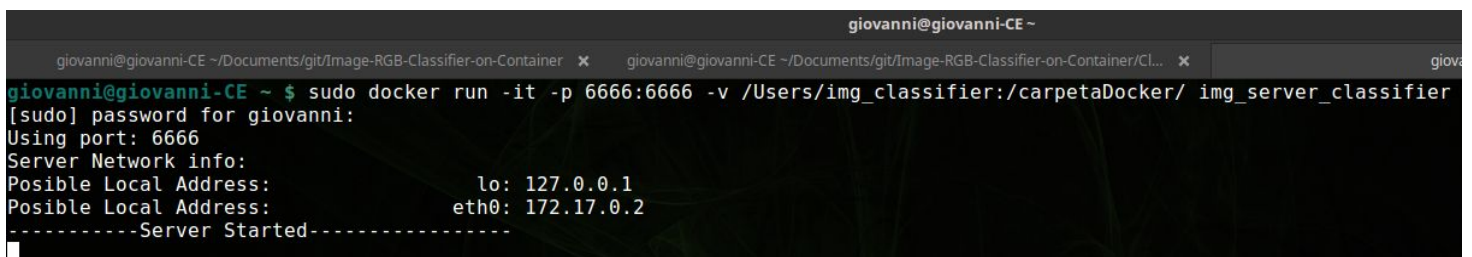
Dentro del comprimido en la carpeta Client hay una carpeta llamada client, dentro de esta carpeta hay un ejecutable para cliente de igual manera y funciona igual al del punto 3.1 y para su ejecución se debe correr

```
./client <ip del host>
```

Esta opción se brinda como un adicional por cualquier error con la manera 3.1

### 3.3 Ejemplo de corrida

En la siguiente imagen se observa cómo se ejecutó el server.



```
giovanni@giovanni-CE ~
giovanni@giovanni-CE ~/Documents/git/Image-RGB-Classifer-on-Container x giovanni@giovanni-CE ~/Documents/git/Image-RGB-Classifer-on-Container/Cl... x giovanni@giovanni-CE ~
giovanni@giovanni-CE ~ $ sudo docker run -it -p 6666:6666 -v /Users/img_classifier:/carpetaDocker/ img_server_classifier
[sudo] password for giovanni:
Using port: 6666
Server Network info:
Possible Local Address:      lo: 127.0.0.1
Possible Local Address:      eth0: 172.17.0.2
-----Server Started-----
```

En la siguiente imagen se obtuvo la ip en esa misma máquina que ejecuto el server con *ifconfig*

```
giovanni@giovanni-CE ~ $ ifconfig
docker0  Link encap:Ethernet  HWaddr 02:42:14:0f:4e:b5
          inet addr:172.17.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::42:14ff:fe0f:4eb5/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:142 errors:0 dropped:0 overruns:0 frame:0
          TX packets:520 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:7304 (7.3 KB)  TX bytes:5884977 (5.8 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1163 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1163 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:100736 (100.7 KB)  TX bytes:100736 (100.7 KB)

veth0b08ebf  Link encap:Ethernet  HWaddr 96:11:4f:8e:b7:a1
             inet6 addr: fe80::9411:4fff:fe8e:b7a1/64  Scope:Link
             UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
             RX packets:0 errors:0 dropped:0 overruns:0 frame:0
             TX packets:50 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:0
             RX bytes:0 (0.0 B)  TX bytes:7101 (7.1 KB)

wlp1s0     Link encap:Ethernet  HWaddr ac:2b:6e:b2:34:cd
             inet addr:192.168.0.103  Bcast:192.168.0.255  Mask:255.255.255.0
             inet6 addr: fe80::e2c6:74b6:1ebf:c0e9/64  Scope:Link
             UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
             RX packets:133487 errors:0 dropped:0 overruns:0 frame:0
             TX packets:84586 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:1000
             RX bytes:169201563 (169.2 MB)  TX bytes:17985679 (17.9 MB)

giovanni@giovanni-CE ~ $
```

En este caso la ip se encuentra en `wlp1s0` puesto que el programa está hecho para no solamente trabajar local si no con otros dispositivos en la misma red, vemos como la ip es `192.168.0.192`.

Ahora procedemos a observar dos clientes de ejemplo, uno que se encuentra en la misma computadora y otro que se encuentra en otra computadora pero en la misma red.



Cliente en misma computadora.

```
giovanni@giovanni-CE ~/Documents/git/Image-RGB-Classifer-on-Container/Client
giovanni@giovanni-CE ~/Documents/git/Image-RGB-Classifer-on-Container/Client $ python client.py 192.168.0.103
Established Connection with Server of ip: 192.168.0.103
Enter image path: 
```

Cliente en distinta computadora.

La siguiente imagen es para demostrar que es una computadora distinta

```
giovanni@giovanni-CE ~/Documents/git/Image-RGB-Classifer-on-Container
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

lo: Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:415 errors:0 dropped:0 overruns:0 frame:0
TX packets:415 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1
RX bytes:36233 (36.2 KB) TX bytes:36233 (36.2 KB)

wlo1: Link encap:Ethernet HWaddr a4:17:31:08:5c:eb
inet addr:192.168.0.109 Bcast:192.168.0.255 Mask:255.255.255.0
inet6 addr: fe80::2b9c:31d:891:9202/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:30051 errors:0 dropped:0 overruns:0 frame:0
TX packets:19506 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:37218388 (37.2 MB) TX bytes:2343130 (2.3 MB)

giovanni@giovanni-CE ~/Documents/git/Image-RGB-Classifer-on-Container $ 
```

```
giovanni@giovanni-CE ~/Documents/git/Image-RGB-Classifer-on-Container/Client
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

lo: Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:415 errors:0 dropped:0 overruns:0 frame:0
TX packets:415 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1
RX bytes:36233 (36.2 KB) TX bytes:36233 (36.2 KB)

wlo1: Link encap:Ethernet HWaddr a4:17:31:08:5c:eb
inet addr:192.168.0.109 Bcast:192.168.0.255 Mask:255.255.255.0
inet6 addr: fe80::2b9c:31d:891:9202/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:30051 errors:0 dropped:0 overruns:0 frame:0
TX packets:19506 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000

giovanni@giovanni-CE ~/Documents/git/Image-RGB-Classifer-on-Container/Client $ python client.py 192.168.0.103
Established Connection with Server of ip: 192.168.0.103
Enter image path: 
```

En las imágenes anteriores puede observarse cómo efectivamente en ambas computadoras la conexión fue exitosa. Seguidamente solo se debe ingresar en el cliente la dirección de la imagen que se desea enviar y el programa indica cuando se terminó de enviar.