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 Operative Systems Principles)

Tarea Corta 2: Container Jobs

(Homework 1: Container Jobs)

Realizado por:

Made by:

Sebastián González Quesada 2013030999

Profesora:

(Professor)

Ing. Alejandra Bolaños Murillo

Fecha: Cartago, Octubre 24, 2017

(Date: October 24th ,2017)

Algoritmo de Color

Este algoritmo funciona calculando el valor [R,G,B] promedio de toda la imagen. A partir de este se hacen comparaciones entre los valores R, G y B para calcular el color predominante. El algoritmo de cálculo del valor promedio del píxel utiliza la biblioteca Pillow y el ejemplo del algoritmo fue tomado de:
http://blog.zeevgilovitz.com/detecting-dominant-colours-in-python/

```
def most_frequent_colour(image):

    w, h = image.size
    pixels = image.getcolors(w * h)

    most_frequent_pixel = pixels[0]

    for count, colour in pixels:
        if count > most_frequent_pixel[0]:
            most_frequent_pixel = (count, colour)
    return most_frequent_pixel
```

Figura 1. Algoritmo pixel promedio

En la figura 1 se ve el algoritmo del píxel promedio basado en el ejemplo ya mencionado.

```
im = Image.open(nameOfFile);
(cont,pixel) = most_frequent_colour(im);
#print pixel;

r = pixel[0];
g = pixel[1];
b = pixel[2];

if (not (validarIp(client_address))): #save in not trusted
        im.save( carpN +"/" + nameOfFile)
elif ((r>g) and (r>b)):
        im.save( carpR +"/" + nameOfFile)
elif((g>r) and (g>b)):
        im.save( carpG +"/" + nameOfFile)
else:
        im.save( carpB +"/" + nameOfFile)
```

Figura 2. Modificaciones

A este se le hicieron las modificaciones respectivas para que se adapte al problema de detectar el color predominante, como las comparaciones mencionadas anteriormente. Se pueden ver en figura 2.

Las carpetas

Primero se montó un volumen llamado *volumen_imagenes* con el siguiente comando:

$ sudo docker volume create volumen_imagenes

Para montar el volumen de un contenedor se utiliza:

$sudo docker run -it -v volumen_imagenes:/carpetaDocker sebas/server:2.0

Donde se crea una carpeta llamada carpetaDocker. Donde estará el archivo de configuración y la carpeta de archivos del contenedor que montó ese volumen. Esto porque varios contenedores pueden montar el mismo volumen. De esta forma se creará dentro de carpetaDocker una carpeta llamada *container1* que es la carpeta de ese contenedor en el volumen.
En la figura 3 se muestra la raíz del sistema de archivos del volumen, y se puede apreciar l*a* capeta *carpetaDocker* mencionada anteriormente.

Figura 3. Sistema de archivos del volumen

Al entrar en *carpetaDocker* se puede observar el archivo de configuración y la carpeta de archivos *container1.*

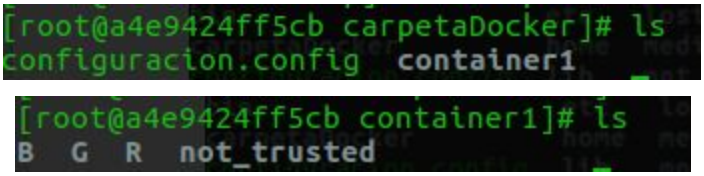


Figura 4. Contenido de Carpetadocker

Dentro de la carpeta de archivos *contaner1* se puede ver las carpetas *R,G,B* y *not_trusted.*

Archivo de Configuración

Figura 5. Ejemplo archivo configuración

El archivo de configuración simplemente tiene n ip's que van a ser aceptadas por el servidor.

Comunicación

La comunicación se da mediante la creación de un socket en python, con la arquitectura cliente-servidor. El código de basa en un ejemplo tomado de:
https://pymotw.com/2/socket/tcp.html
El servidor se ejecuta en el contenedor, mientras que el cliente fuera del contenedor.

Puerto a Utilizar

Se utiliza el puerto 10000

Programa que envía las imágenes

Es programa que envía imágenes es un cliente implementado en python, que recibe como parámetro la dirección ip donde se enviarán las imágenes.

```
import socket
import sys
ip =sys.argv[1]
print "Socket establecido con HOST1"
while True:
        # Create a TCP/IP socket
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        # Connect the socket to the port where the server is listening
        server_address = (ip, 10000)
        sock.connect(server_address)
        try:
                print "Ingrese la ruta de la imagen"
                ruta = raw_input("")
                sock.sendall(ruta)
                if (ruta == "salir"):
                        sock.sendall("salir")
                        break
                file = open(ruta, "rb")
                # Send data
                content = file.read(1024)
                while True:
                        if content:
                                sock.sendall(content)
                                content = file.read(1024)
                        else:
                                print "Enviando imagen ........"
                                break
                print "Imagen enviada"
                file.close()
        finally:
            sock.close()
```

Figura 6 . Programa que envía imágenes

Dockerfile

```
FROM centos:7
RUN yum install -y gcc-c++
RUN yum provides ifconfig
RUN yum install -y net-tools
RUN yum install -y libjpeg-devel
RUN yum install -y python-imaging numpy
ADD server.py ./
ADD configuracion.config ./
```

Figura 7. Dockerfile

Guía de Usuario

1) Entrar en la carpeta Tarea2_SebastianGonzalez
2) Ingresas la ip de la computadora que se utilizará como cliente en el archivo *configuracion.config*, de otra forma todas las imágenes serán clasificadas como *not_trusted.* En el archivo de configuración ya está la ip 172.17.0.1 que es la ip por defecto del cliente (programa que envía imágenes).
3) Ejecutar el comando, para crear un volumen:

   $sudo docker volume create volumen_imagenes
4) Abrir una terminal en la carpeta del punto 1 y ejecutar:

   $ sudo docker build -t sebas/server:2.0 .
5) Ejecutar:

   $ sudo docker run -it -v volumen_imagenes:/carpetaDocker sebas/server:2.0

   Esto desplegará la siguiente información

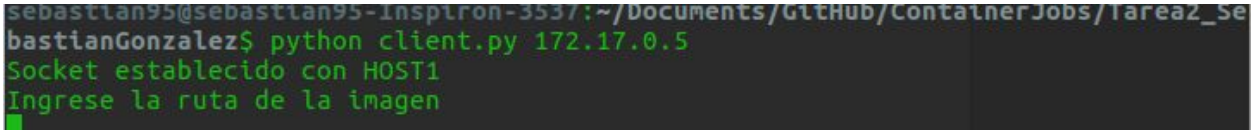
```
sebastian95@sebastian95-Inspiron-3537:~/Documents/GitHub/ContainerJobs/Tarea2_Se
bastianGonzalez$ sudo docker run -it -v volumen_imagenes:/carpetaDocker sebas/se
rver:2.0
Ip: 172.17.0.5
iniciando socket con HOST2
```

   Donde Ip es la ip del servidor (donde hay que enviar las imágenes).
6) Después de esto sigue ejecutar el cliente. Para esto se debe abrir otra terminal en la carpeta Tarea2_SebastianGonzalez y ejecutar el comando:

   $ python client.py <ip>

   Ingresando la ip de la imagen anterior.

7) Se desplegará en la consola la siguiente información:

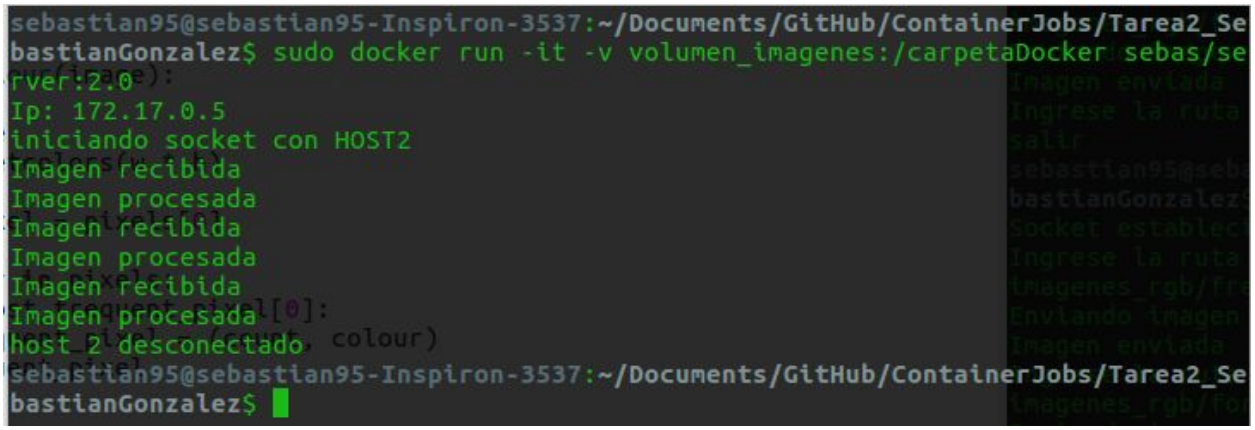


En este punto se debe ingresar la ruta de una imagen, en este caso se pueden ingresar rutas relativas o absolutas, sin embargo es preferible que sean absolutas.

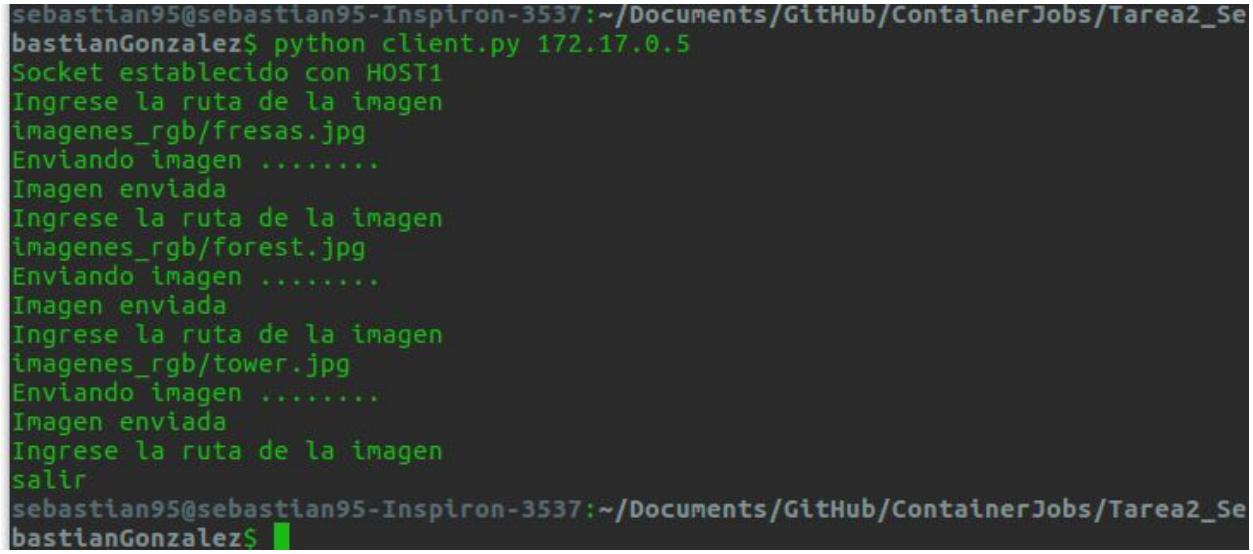
8) Para cerrar la conexión se debe escribir salir.

Ejemplo de corrida:

Terminal de servidor



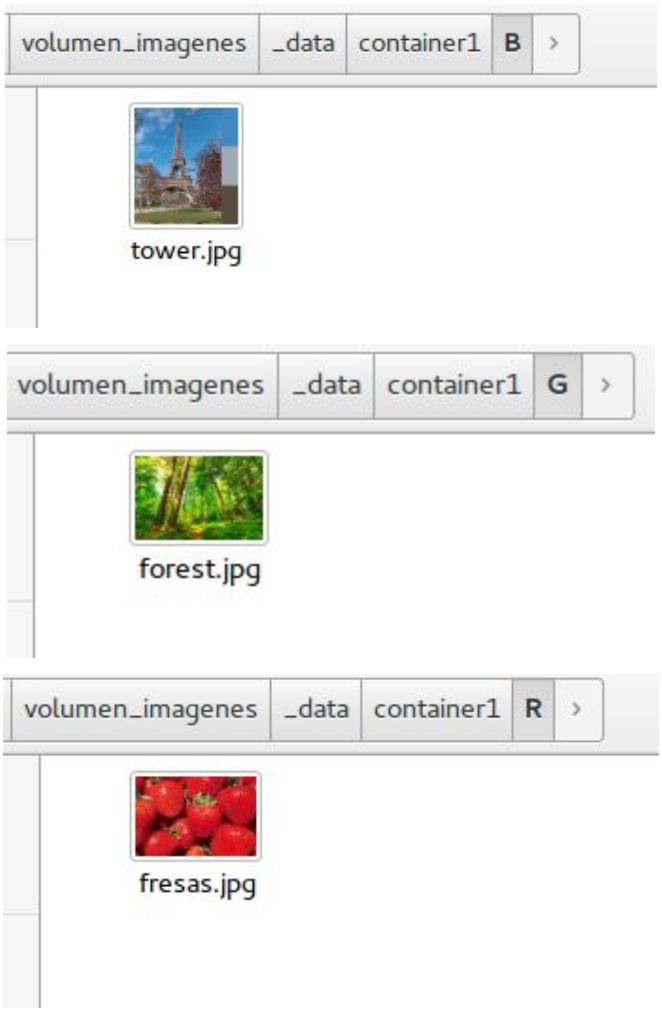
Terminal de cliente



Los resultados se pueden ver, con los comandos
$ su

cd /var/lib/docker/volumes/volumen_imagenes/_data

#nautilus ./
Se entra en la carpeta container1 con el explorador de archivos y entrando en las carpetas se ven los resultados.



En la carpeta not_trusted no hay imágenes dado que el ejemplo es un una ip registrada en el archivo de configuración.

Referencias:

1) https://www.digitalocean.com/community/tutorials/como-instalar-y-usar-docker-en-ubuntu-16-04-es

2) http://www.pythonforbeginners.com/files/reading-and-writing-files-in-python