

Computación en la Nube Deber 1

-Elian Garcia

Homework 1 - Containers and Virtual Machines

In this assignment, we will be comparing the usage and performance of both virtual machines and containers. For this activity, you'll be working on groups of 2 people (max 3).

Requirements:

- The server: 1 computer running Vagrant and Docker
- The client: 1 computer with at least Vagrant

Experiment 1:

For the server, write a Vagrant file that starts a Ubuntu VM, installs and configures a web server such as NGINX, and starts up the web server. The web server should host a very simple web page.

Client .Vagrant

En primer lugar, vamos a empezar con el cliente de Vagrant, ya que este está encargado de correr el "wrk" test para poder ejercer a las máquinas virtuales de Docker y de Vagrant un alto nivel de tráfico HTTP.

Con el fin de crear un .vagrant file que funcione en la distribución de Linux designada, corremos en el cmd, en la ubicación que desee tener las configuraciones de la máquina virtual, el siguiente comando:

```
vagrant init hashicorp/bionic64
```

Una vez creados los archivos respectivos para poder correr esta distribución, reemplazamos el .vagrant file por la siguiente configuración:

```
# Vagrantfile
Vagrant.configure("2") do |config|
```

```

config.vm.box = "hashicorp/bionic64"

config.vm.provision "shell", inline: <<-SHELL
  apt update
  apt install -y build-essential git unzip

  # Clone the wrk repository
  git clone https://github.com/wg/wrk.git

  # Change directory to wrk and build it
  cd wrk
  make

  # Move the compiled wrk binary to /usr/local/bin
  mv wrk /usr/local/bin

  # Check wrk version
  wrk --version

  # Run wrk against a test server

SHELL
end

```

Ahora corremos el comando `vagrant up` para activar la máquina virtual. Una vez finalice, se tendrá el servicio de "wrk" instalado correctamente y listo para realizar las pruebas respectivas.

La sintaxis de "wrk" funciona de la siguiente manera:

```

-c, --connections: total number of HTTP connections to keep open
                    each thread handling N = connections/threads

-d, --duration:     duration of the test, e.g. 2s, 2m, 2h

```

```
-t, --threads:      total number of threads to use

-s, --script:       LuaJIT script, see SCRIPTING

-H, --header:       HTTP header to add to request, e.g. "User-Agent"

--latency:          print detailed latency statistics

--timeout:          record a timeout if a response is not received
                    this amount of time.
```

Instalación Server Vagrant

Vamos a crear un archivo .vagrant con la misma distribución escogida para el cliente.

```
vagrant init hashicorp/bionic64
```

Una vez creados los archivos respectivos para poder correr esta distribución, reemplazamos el archivo .vagrant por la siguiente configuración:

Server .vagrant file

```
config.vm.provision "shell", inline: <<-SHELL

#instalar nginx

apt-get
apt-get install -y nginx

#enable nginx
systemctl start nginx
systemctl enable nginx

#Creacion de la pagina web
echo      '<!DOCTYPE html>'
          '<html lang="es">
```

```

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-w:
  <title>Computación en la Nube deber 1</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f0f0f0;
      margin: 0;
      padding: 20px;
    }
    h1 {
      color: #333;
    }
    p {
      font-size: 18px;
    }
    .content {
      background-color: #fff;
      padding: 20px;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }
  </style>
</head>
<body>
  <div class="content">
    <h1>Computación en la Nube - Deber 1</h1>
    <p>Elían García.</p>
  </div>
</body>
</html>' | sudo tee /var/www/html/index.nginx-deb:

```

#permisos

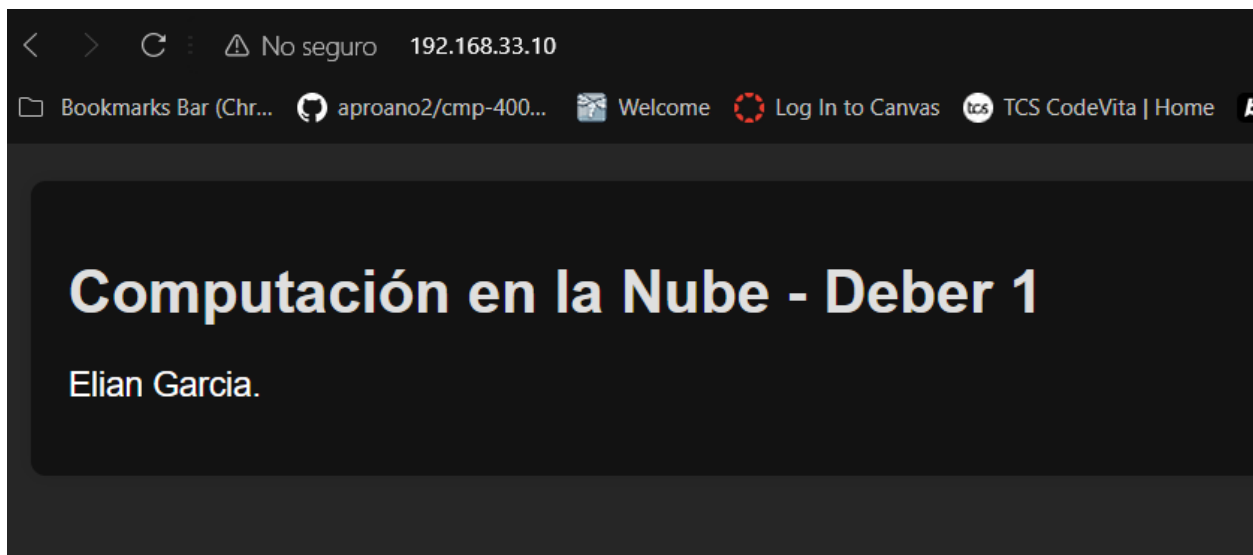
```
sudo chown -R www-data:www-data /var/www/html
```

```
SHELL
config.vm.network "private_network", ip: "192.168.33.10"
config.vm.network "forwarded_port", guest: 80, host: 8080

end
```

Ahora vamos a ejecutar `vagrant up` en la carpeta donde se encuentra el documento ".vagrant". Una vez ejecutado el comando, esta máquina virtual tendrá todas las características que especificamos en el documento. Cabe recalcar que en el documento .vagrant se configuró la red de la máquina virtual dándole una IP y posteriormente habilitando los puertos para el tráfico HTTP.

Si accedemos a la dirección <http://192.168.33.10> podemos encontrar la página publicada:



Se utiliza el paquete de htop para poder analizar el rendimiento de nuestra máquina virtual:

```
vagrant@vagrant:~$ htop
```

Rendimiento antes de test:

CPU[|0.7%]

Mem[|||||76.5M/985M]

Swp[0K/980M]

Tasks: 27, 18 thr; 1 running
Load average: 0.00 0.00 0.00
Uptime: 00:28:26

| PID | USER | PRI | NI | VIRT | RES | SHR | S | CPU% | MEM% | TIME+ | Command |
|------|-----------|-----|----|-------|-------|-------|---|------|------|---------|---------------|
| 1043 | root | 20 | 0 | 369M | 2608 | 2244 | S | 1.3 | 0.3 | 0:00.60 | /usr/sbin/VBo |
| 1 | root | 20 | 0 | 77648 | 8720 | 6600 | S | 0.0 | 0.9 | 0:01.22 | /sbin/init |
| 375 | root | 19 | -1 | 94792 | 14504 | 13800 | S | 0.0 | 1.4 | 0:00.18 | /lib/systemd/ |
| 398 | root | 20 | 0 | 46620 | 5648 | 3272 | S | 0.0 | 0.6 | 0:00.47 | /lib/systemd/ |
| 399 | root | 20 | 0 | 103M | 1932 | 1704 | S | 0.0 | 0.2 | 0:00.00 | /sbin/lvm |
| 429 | root | 20 | 0 | 47600 | 3624 | 3224 | S | 0.0 | 0.4 | 0:00.01 | /sbin/rpcbind |
| 457 | systemd-r | 20 | 0 | 70628 | 5248 | 4688 | S | 0.0 | 0.5 | 0:00.03 | /lib/systemd/ |
| 553 | root | 20 | 0 | 31320 | 3196 | 2920 | S | 0.0 | 0.3 | 0:00.00 | /usr/sbin/cro |
| 556 | root | 20 | 0 | 70608 | 5896 | 5160 | S | 0.0 | 0.6 | 0:00.04 | /lib/systemd/ |
| 559 | messagebu | 20 | 0 | 50060 | 4268 | 3796 | S | 0.0 | 0.4 | 0:00.10 | /usr/bin/dbus |

Rendimiento despues de test:

Se realiza el test de "wrk" accediendo al cliente mediante `vagrant ssh` y posteriormente corremos el siguiente comando para poder ejercer carga al servidor donde esta publicado nuestra pagina web:

```
wrk -c300 -d3m -t15 http://192.168.33.10
```

Servidor sin carga:

| | | |
|------|-------------|------------------------------|
| CPU[| 0.7%] | Tasks: 27, 18 thr; 1 running |
| Mem[| 76.8M/985M] | Load average: 0.06 0.26 0.15 |
| Swp[| 0K/980M] | Uptime: 00:34:39 |

| PID | USER | PRI | NI | VIRT | RES | SHR | S | CPU% | MEM% | TIME+ | Command |
|------|-----------|-----|----|-------|-------|-------|---|------|------|---------|---------------|
| 1729 | vagrant | 20 | 0 | 105M | 5404 | 4260 | S | 0.7 | 0.5 | 0:00.37 | sshd: vagrant |
| 1 | root | 20 | 0 | 77648 | 8720 | 6600 | S | 0.0 | 0.9 | 0:01.22 | /sbin/init |
| 375 | root | 19 | -1 | 94792 | 14504 | 13800 | S | 0.0 | 1.4 | 0:00.18 | /lib/systemd/ |
| 398 | root | 20 | 0 | 46620 | 5648 | 3272 | S | 0.0 | 0.6 | 0:00.47 | /lib/systemd/ |
| 399 | root | 20 | 0 | 103M | 1932 | 1704 | S | 0.0 | 0.2 | 0:00.00 | /sbin/lvm |
| 429 | root | 20 | 0 | 47600 | 3624 | 3224 | S | 0.0 | 0.4 | 0:00.01 | /sbin/rpcbind |
| 457 | systemd-r | 20 | 0 | 70628 | 5248 | 4688 | S | 0.0 | 0.5 | 0:00.03 | /lib/systemd/ |

Minuto 1:

```

CPU[|||||100.0%] Tasks: 27, 18 thr; 1 running
Mem[|||||81.0M/985M] Load average: 0.39 0.10 0.03
Swp[|||||0K/980M] Uptime: 00:29:22

  PID USER      PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
  658 www-data   20    0  140M  7256  5300  R   92.7  0.7   0:33.13 nginx: worker process
 1749 vagrant    20    0 27140  4292  3364  R   0.0  0.4   0:00.02 htop
 1729 vagrant    20    0  105M  5404  4260  S   0.0  0.5   0:00.13 sshd: vagrant@pts/0
 1043 root        20    0  369M  2608  2244  S   0.0  0.3   0:00.62 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-service.sh
    1 root        20    0 77648  8720  6600  S   0.0  0.9   0:01.22 /sbin/init
   375 root       19   -1 94792 14504 13800  S   0.0  1.4   0:00.18 /lib/systemd/systemd-journald

```

Ultimo minuto:

```

CPU[|||||100.0%] Tasks: 27, 18 thr; 1 running
Mem[|||||81.9M/985M] Load average: 0.93 0.42 0.16
Swp[|||||0K/980M] Uptime: 00:31:34

  PID USER      PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
  658 www-data   20    0  140M  7256  5300  R   93.3  0.7  2:35.83 nginx: worker process
 1749 vagrant    20    0 27140  4292  3364  R   0.7  0.4   0:00.18 htop
 1043 root        20    0  369M  2608  2244  S   0.7  0.3   0:00.67 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-service.sh
 1060 root        20    0  369M  2608  2244  S   0.7  0.3   0:00.42 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-service.sh
 1729 vagrant    20    0  105M  5404  4260  S   0.0  0.5   0:00.28 sshd: vagrant@pts/0
 1057 root        20    0  369M  2608  2244  S   0.0  0.3   0:00.19 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-service.sh
 559 messagebu  20    0 50060  4268  3796  S   0.0  0.4   0:00.11 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopid
    1 root        20    0 77648  8720  6600  S   0.0  0.9   0:01.22 /sbin/init
   375 root       19   -1 94792 14504 13800  S   0.0  1.4   0:00.18 /lib/systemd/systemd-journald
   398 root        20    0  46620  5648  3272  S   0.0  0.6   0:00.47 /lib/systemd/systemd-udev
   399 root        20    0  103M  1932  1704  S   0.0  0.2   0:00.00 /sbin/lvmtool -f
 429 root        20    0  47600  3624  3224  S   0.0  0.4   0:00.01 /sbin/rachid -f -w

```

Estadísticas de test:

```

vagrant@vagrant:~$ wrk -c300 -d3m -t15 http://192.168.33.10
Running 3m test @ http://192.168.33.10
15 threads and 300 connections
  Thread Stats   Avg      Stdev     Max   +/-  Stdev
    Latency    71.25ms   46.48ms   1.98s   90.34%
    Req/Sec   115.60    48.77   430.00   71.67%
306677 requests in 3.00m, 441.62MB read
Socket errors: connect 0, read 0, write 42, timeout 2787
Requests/sec: 1702.89

```

Performance Vagrant Client and Server

Experiment 2:

You'll repeat the same experiment as before. The only difference is that you have to run the web server in Docker instead of a virtual machine.

Docker

Creamos un .dockerfile en la carpeta deseada y configuramos el archivo con :

```
FROM ubuntu:18.04

# paquetes y actualizaciones requeridas
RUN apt update && apt install -y vim htop nginx curl net-tools :

# Agregar nuestra pagina web al root
RUN echo '<!DOCTYPE html> <html lang="es"><head>      <meta charse

#establecer permisos
RUN chown -R www-data:www-data /var/www/html

# Escuchar el puerto 80 para poder tener trafico
EXPOSE 80

# activar el servicio de nginx en segundo plano
CMD ["nginx", "-g", "daemon off;"]
```

Generacion de imagen mediante `docker build -t webserverimage -f .\webserver.dockerfile .`

```
PS C:\Docker> docker build -t webserverimage -f .\webserver.dockerfile .
[+] Building 2.7s (9/9) FINISHED                                docker:desktop-linux
=> [internal] load build definition from webserver.dockerfile    0.0s
=> => transferring dockerfile: 1.26kB                             0.0s
=> [internal] load metadata for docker.io/library/ubuntu:18.04  1.0s
=> [auth] library/ubuntu:pull token for registry-1.docker.io    0.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                       0.0s
=> [1/4] FROM docker.io/library/ubuntu:18.04@sha256:152dc042452c496007f07ca9127571cb9c29697f42acbfad72324b2bb2e4 0.0s
=> => resolve docker.io/library/ubuntu:18.04@sha256:152dc042452c496007f07ca9127571cb9c29697f42acbfad72324b2bb2e4 0.0s
=> CACHED [2/4] RUN apt update && apt install -y vim htop nginx curl net-tools iptables 0.0s
=> CACHED [3/4] RUN echo '<!DOCTYPE html> <html lang="es"><head>      <meta charset="UTF-8">      <meta name="viewpo 0.0s
=> CACHED [4/4] RUN chown -R www-data:www-data /var/www/html    0.0s
=> exporting to image                                           1.5s
=> => exporting layers                                             0.0s
=> => exporting manifest sha256:b14329a7874ed3e1158bf6f9f8890876a8de35c50bedc957005c22e7d1796253 0.0s
=> => exporting config sha256:d9ab9cde5e6acf2c2a7f0600e50ec4b7be900f9c77d52be8b4b54bef6fb9b587 0.0s
=> => exporting attestation manifest sha256:11feab558e0536c020d8c14eeffa512befe7c18ab0a674e5abec45d902493f189 0.0s
=> => exporting manifest list sha256:f589a5cac4cd7de6bf512812b7228bfa943dbe6cbe591c2bbcb124bef7803a700 0.0s
=> => naming to docker.io/library/webserverimage:latest         0.0s
=> => unpacking to docker.io/library/webserverimage:latest      1.4s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/yrgf98gtrvxv37i3h19v6orxp

What's next:
  View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Docker>
```


Generacion de container mediante `docker run -d -p 80:80 mynginx :`

```
PS C:\Docker> docker run -d -p 80:80 webserverimage
eec6f69daa95e6244de94f9767171b44c87a5d4ca7ee65de48090ddb4cbe71c4
PS C:\Docker> |
```

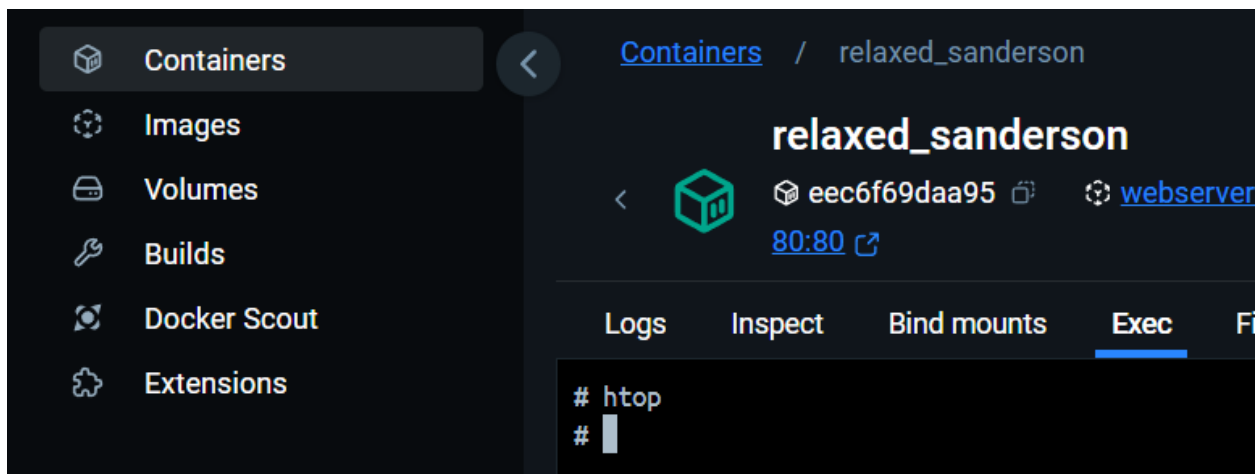
En el lado del **CLIENTE** generamos un “wrk” test el cual tiene los mismos argumentos que el anterior test del experimento 1. El ip al cual el wrk test tiene que generar trafico es al del host del docker.

Se corre el comando:

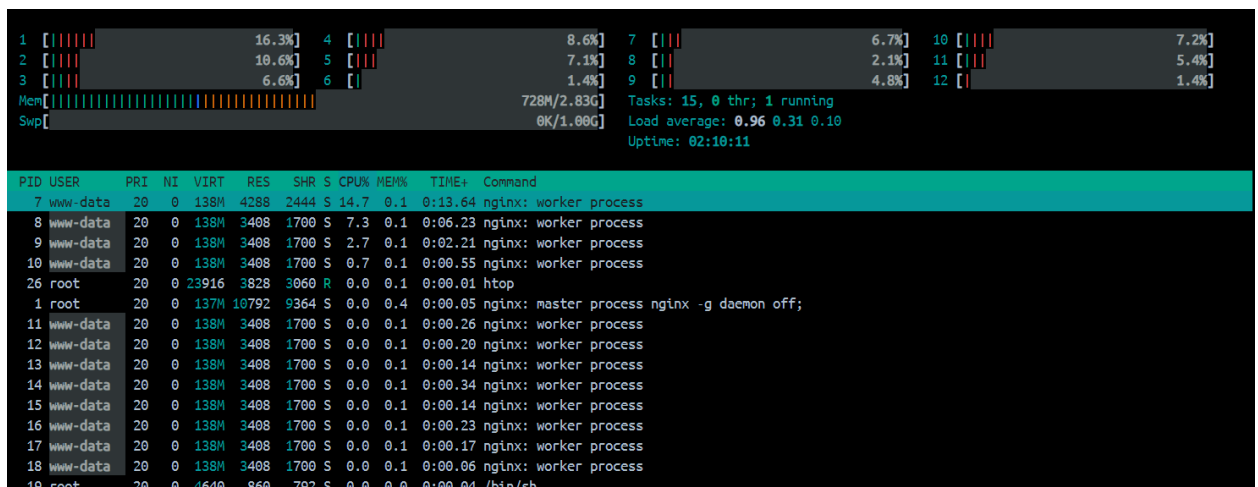
```
wrk -c300 -d3m -t15 http:// 172.23.64.1
```

```
vagrant@vagrant:~$ wrk -c300 -d3m -t15 http://172.23.64.1
Running 3m test @ http://172.23.64.1
 15 threads and 300 connections
   Thread Stats   Avg      Stdev     Max    +/-  Stdev
   Latency    64.88ms   36.65ms  752.51ms   66.19%
   Req/Sec    187.73    55.82   544.00    71.10%
 500496 requests in 3.00m, 484.45MB read
Requests/sec:  2779.55
Transfer/sec:    2.69MB
```

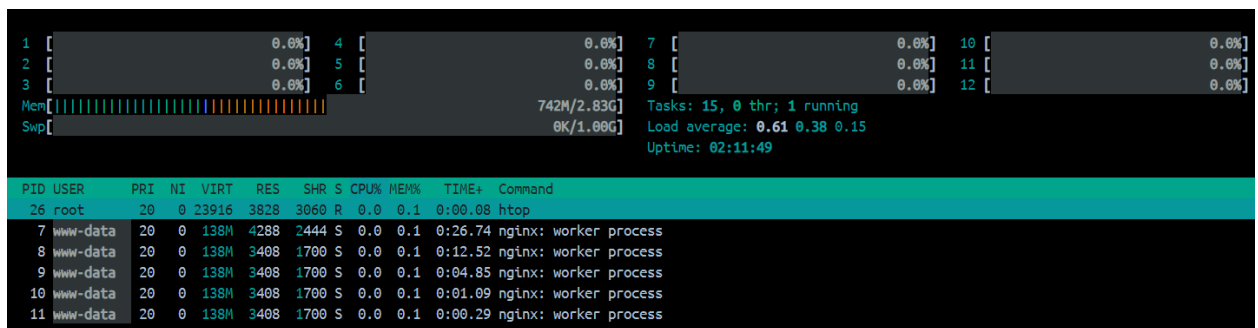
En docker accedemos al container creado y a la pestaña de exec y ejecutamos `htop` para poder examinar el rendimiento utilizando docker.



Observamos el rendimiento:



Una vez finalizado :



Resultados

Expermiento 1

Minuto 1:

```

CPU[|||||]|||||100.0%] Tasks: 27, 18 thr; 1 running
Mem[|||||]81.0M/985M Load average: 0.39 0.10 0.03
Swp[|||||]0K/980M Uptime: 00:29:22

PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
658 www-data   20    0 140M   7256   5300 R 92.7  0.7   0:33.13 nginx: worker process
1749 vagrant    20    0 27140  4292   3364 R  0.0  0.4   0:00.02 httpd
1729 vagrant    20    0 105M   5404   4260 S  0.0  0.5   0:00.13 sshd: vagrant@pts/0
1043 root        20    0 369M   2608   2244 S  0.0  0.3   0:00.62 /usr/sbin/VBoxService --pidfile /var/run/vboxadd-service.sh
1 root        20    0 77648  8720   6600 S  0.0  0.9   0:01.22 /sbin/init
375 root        19   -1 94792 14504 13800 S  0.0  1.4   0:00.18 /lib/systemd/systemd-journald

```

Ultimo minuto:

[illegible]

Experimento 2

Observamos el rendimiento:

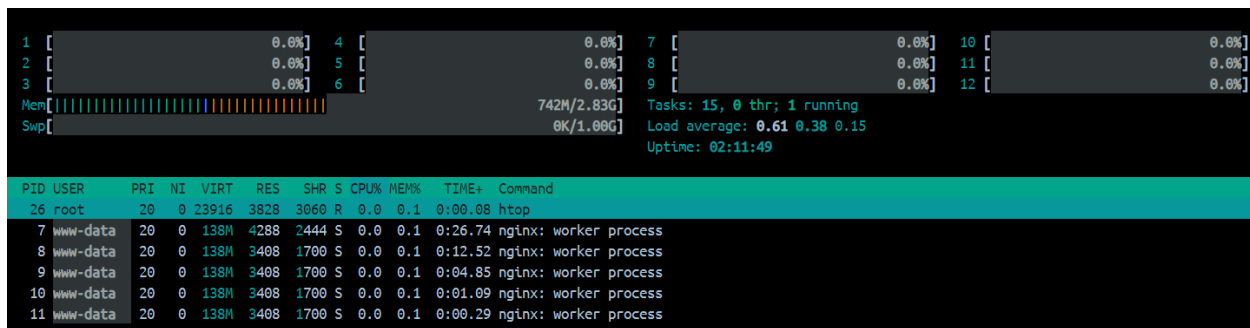
```

1 [|||||] 16.3% 4 [||||] 8.6% 7 [|||] 6.7% 10 [||||] 7.2%
2 [||||] 10.6% 5 [||||] 7.1% 8 [||] 2.1% 11 [||||] 5.4%
3 [||||] 6.6% 6 [||] 1.4% 9 [||] 4.8% 12 [||] 1.4%
Mem[|||||||||||||||||] 728M/2.83G Tasks: 15, 0 thr; 1 running
Swp[|] 0K/1.00G Load average: 0.96 0.31 0.10
Uptime: 02:10:11

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
7 www-data 20 0 138M 4288 2444 S 14.7 0.1 0:13.64 nginx: worker process
8 www-data 20 0 138M 3408 1700 S 7.3 0.1 0:06.23 nginx: worker process
9 www-data 20 0 138M 3408 1700 S 2.7 0.1 0:02.21 nginx: worker process
10 www-data 20 0 138M 3408 1700 S 0.7 0.1 0:00.55 nginx: worker process
26 root 20 0 23916 3828 3060 R 0.0 0.1 0:00.01 htop
1 root 20 0 137M 10792 9364 S 0.0 0.4 0:00.05 nginx: master process nginx -g daemon off;
11 www-data 20 0 138M 3408 1700 S 0.0 0.1 0:00.26 nginx: worker process
12 www-data 20 0 138M 3408 1700 S 0.0 0.1 0:00.20 nginx: worker process
13 www-data 20 0 138M 3408 1700 S 0.0 0.1 0:00.14 nginx: worker process
14 www-data 20 0 138M 3408 1700 S 0.0 0.1 0:00.34 nginx: worker process
15 www-data 20 0 138M 3408 1700 S 0.0 0.1 0:00.14 nginx: worker process
16 www-data 20 0 138M 3408 1700 S 0.0 0.1 0:00.23 nginx: worker process
17 www-data 20 0 138M 3408 1700 S 0.0 0.1 0:00.17 nginx: worker process
18 www-data 20 0 138M 3408 1700 S 0.0 0.1 0:00.06 nginx: worker process
19 root 20 0 640 360 702 S 0.0 0.0 0:00.01 /bin/sh

```

Una vez finalizado :



Conclusiones

En base a los dos experimentos podemos analizar que la utilización de máquinas virtuales es mucho más demandante en varios ámbitos. La máquina virtual tiene mucha más demanda en su CPU, memoria e Input y Output, lo cual se puede ver en el gráfico de "htop" que tiene cerca del 100% de demanda al CPU. Mientras que en el gráfico htop del docker la demanda era alrededor del 17%. Se puede ver claramente las ventajas de utilizar docker en cuanto al CPU. Por otra parte si es que analizamos el rendimiento de la memoria, hay un nivel mucho mas alto en el rendimiento del Vagrant que en el cliente de Docker.

Nginx ejecutándose en Docker tiene menos sobrecarga que Nginx en una máquina virtual debido a la virtualización completa del sistema operativo en las VMs. Los contenedores Docker son más livianos y tienen menor sobrecarga, lo que les permite ofrecer un mejor rendimiento, especialmente en cargas de trabajo intensivas en I/O y red. Por lo tanto, Nginx en Docker ofrecerá un rendimiento superior comparado con Nginx en una máquina virtual en base a este experimento.