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 oper
                   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-Age

    --latency:     print detailed latency statistics

    --timeout:     record a timeout if a response is not receive
                   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

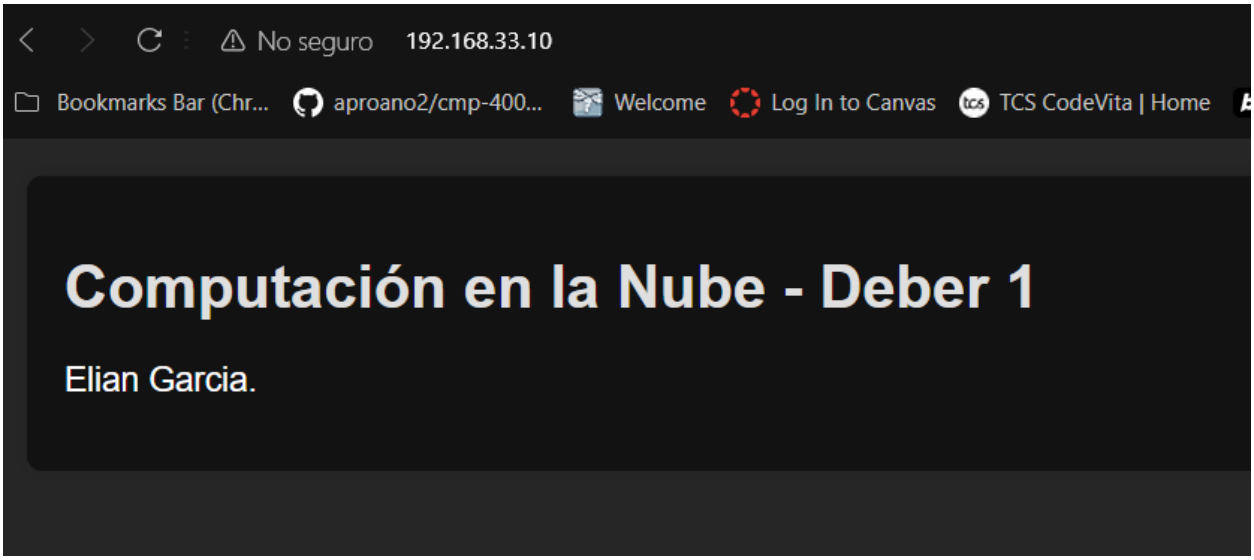
```
config.vm.provision "shell", inline: <<-SHELL

   #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-wi
                <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>Elian Garcia.</p>
                </div>
            </body>
            </html>' | sudo tee /var/www/html/index.nginx-debi

#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.

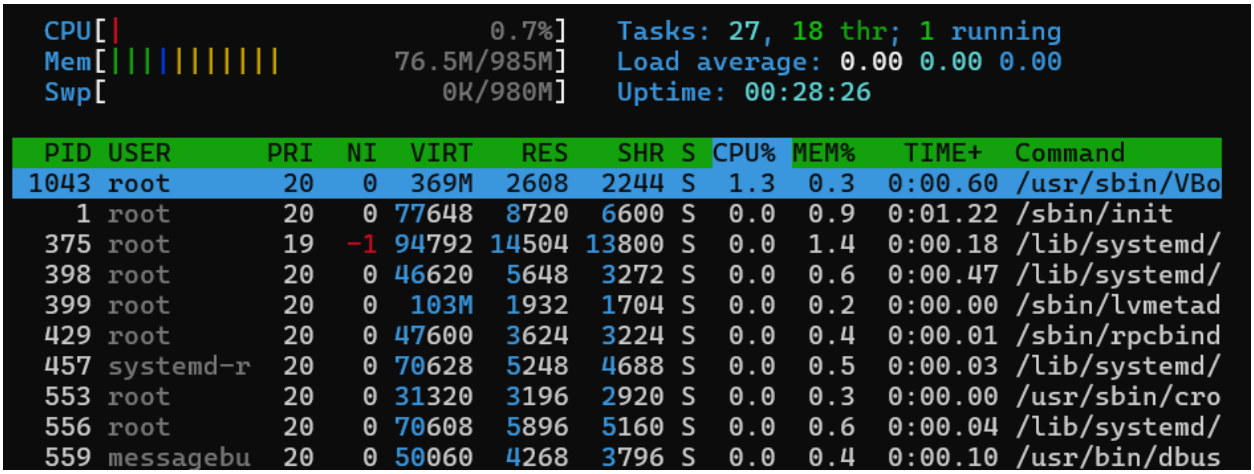
Si accedemos a la direccion http://192.168.33.10 podemos encontrar la página publicada:



Se utiliza el packete de htop para poder analizar el rendimiento de nuestra maquina virtual:

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

Rendimiento antes de test:

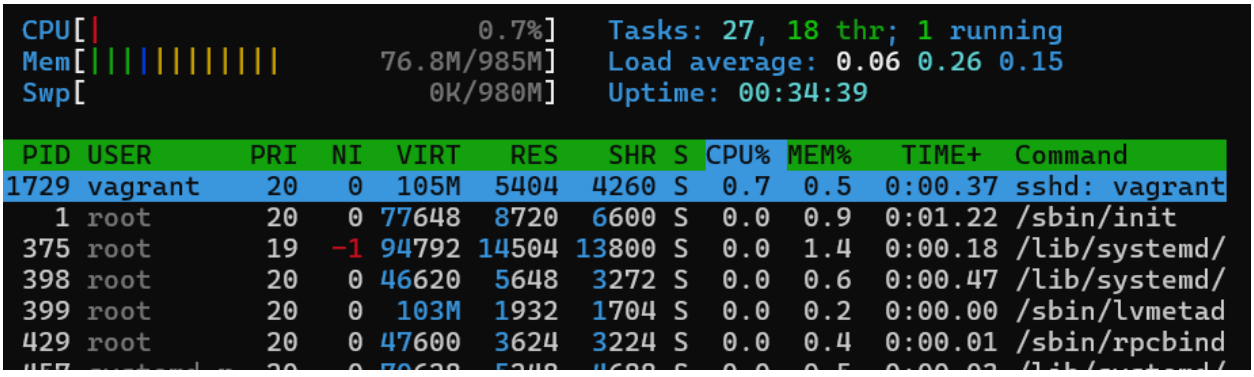


Rendimiento despues de test:

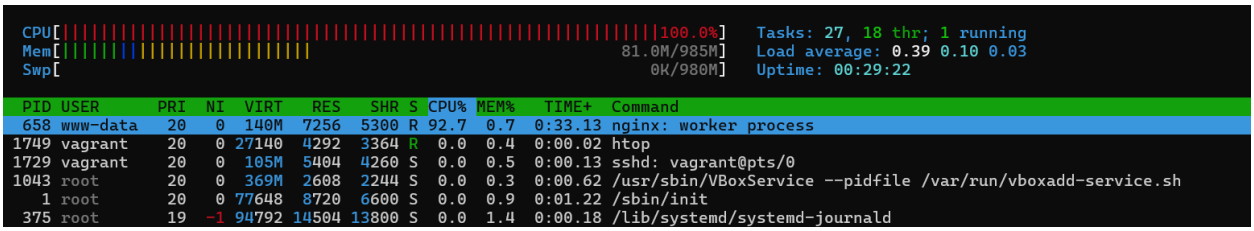
Se realiza el test de “wrk” accediendo al cliente medinate `vagrant ssh` y posteriormente corremos el siguiente comando para poder ejercer carga al servidor donde esta publicado nuestra pagina web:

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

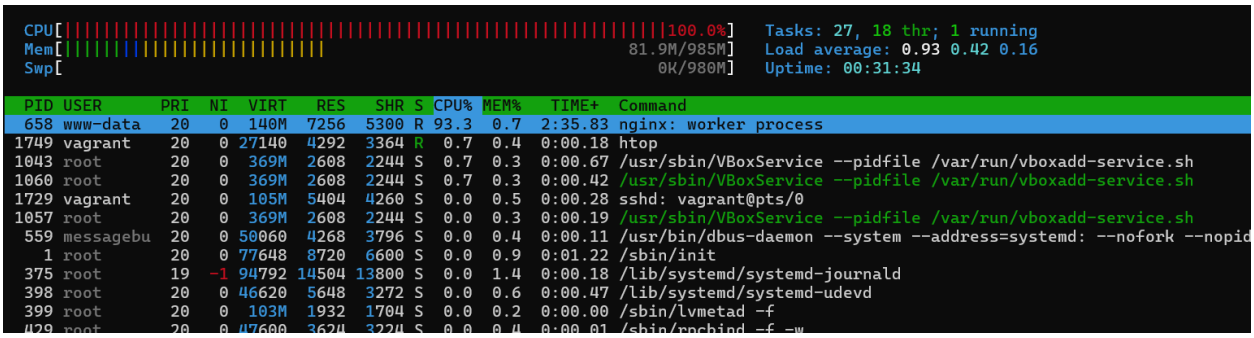
Servidor sin carga:



Minuto 1:



Ultimo minuto:



Estadisticas 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:

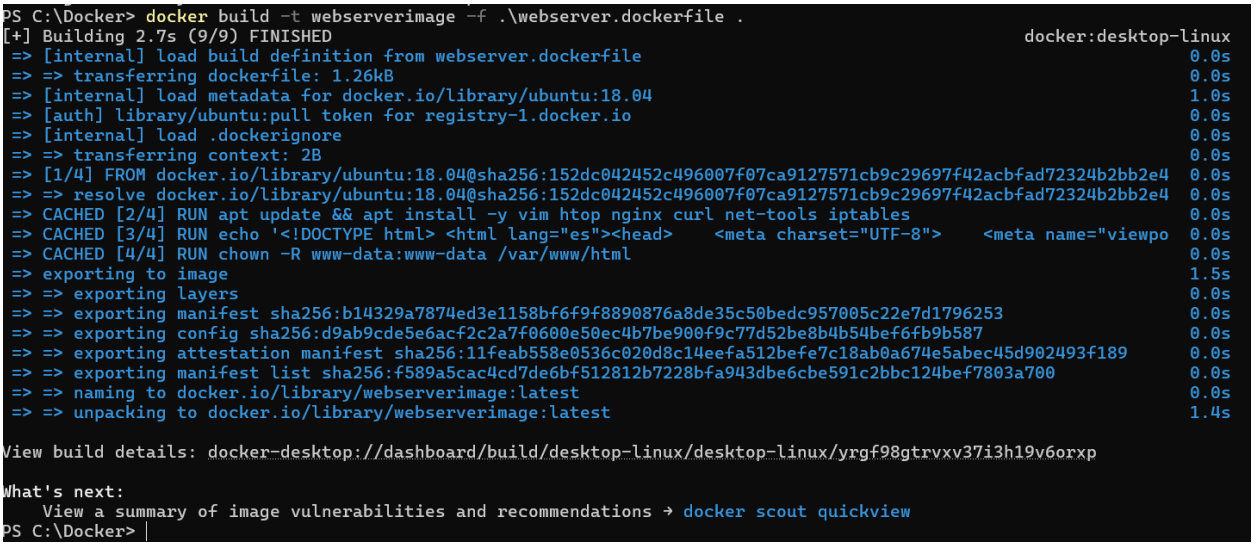
You'll repeat the same experiment as before. The only difference is that you have to run the web server in Docker instead of a virtual machine.

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 .



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>
```

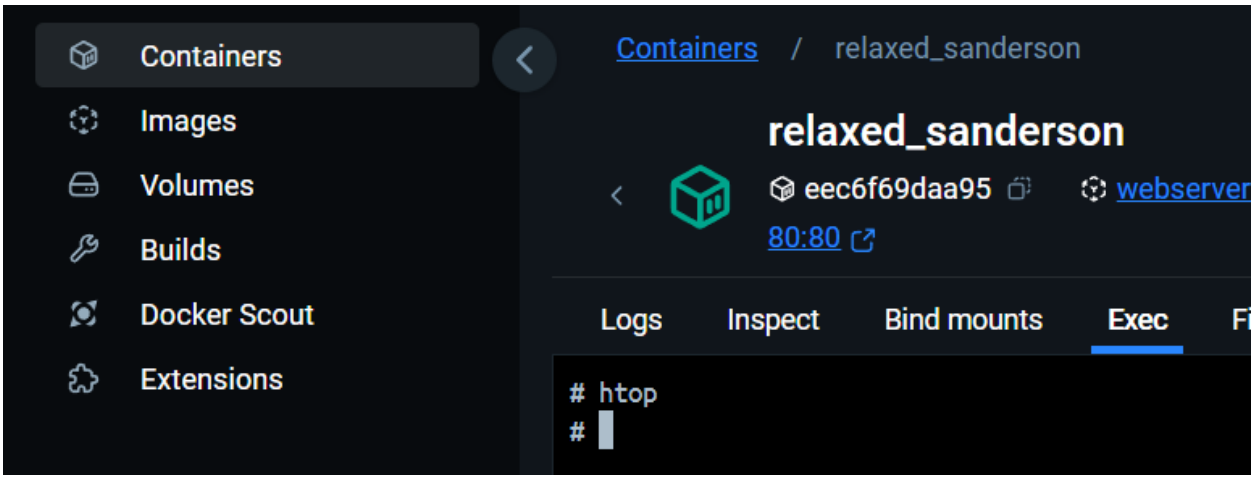
En el lado del **CLIENTE** generamos un “wrk” test el cual tiene los mismos argumentos que el anterior test del experimento 1. El ip al cual el wrk test tiene que generar trafico es al del host del docker.

Se corre el comando:

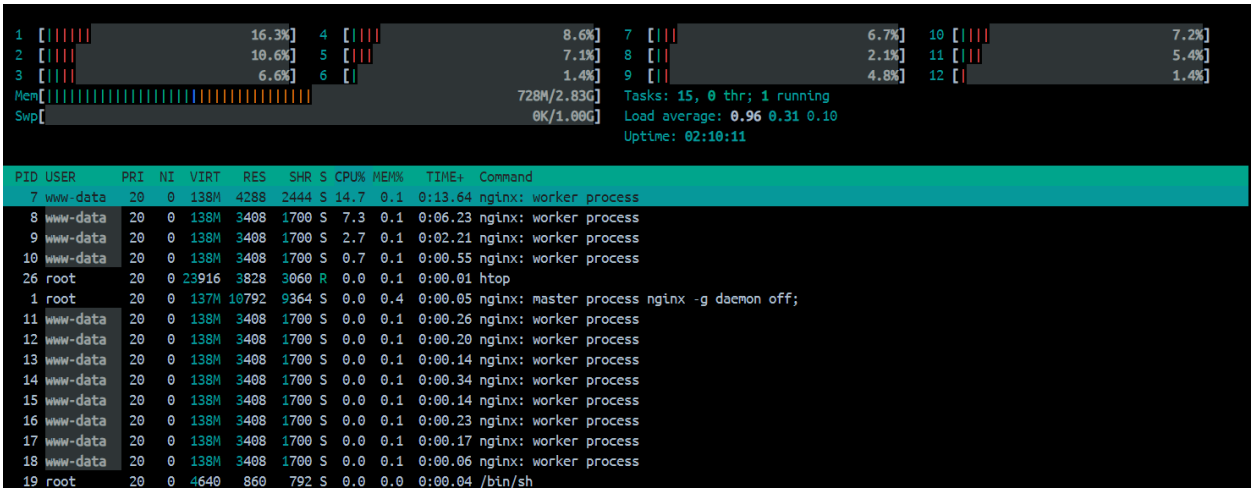
`wrk -c300 -d3m -t15 http://172.23.64.1`

```
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
```

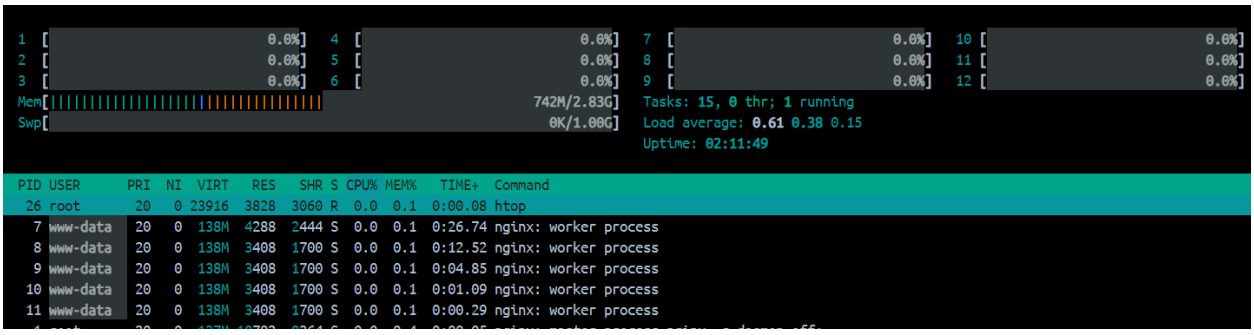
En docker accedemos al container creado y a la pestaña de *exec* y ejecutamos `htop` para poder examinar el rendimiento utilizando docker.



Observamos el rendimiento:



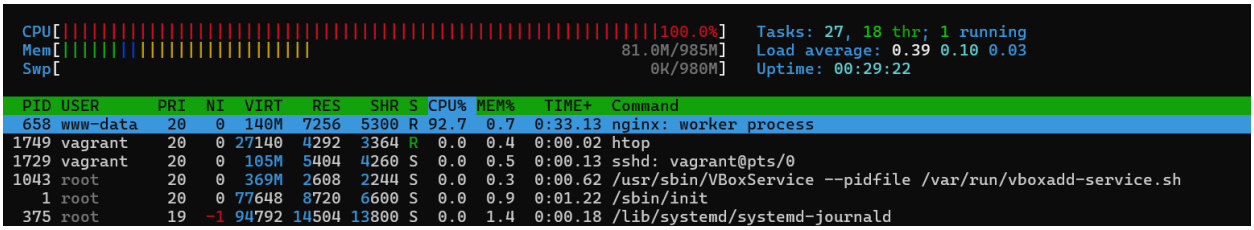
Una vez finalizado :



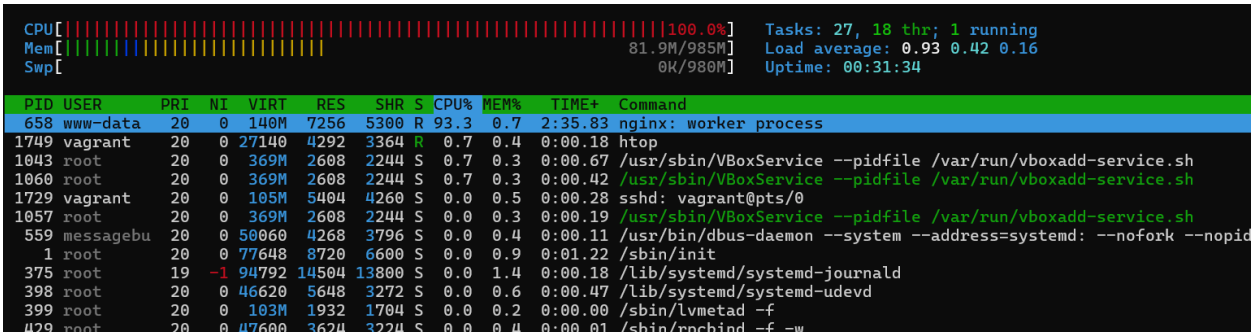
Resultados

Expermiento 1

Minuto 1:

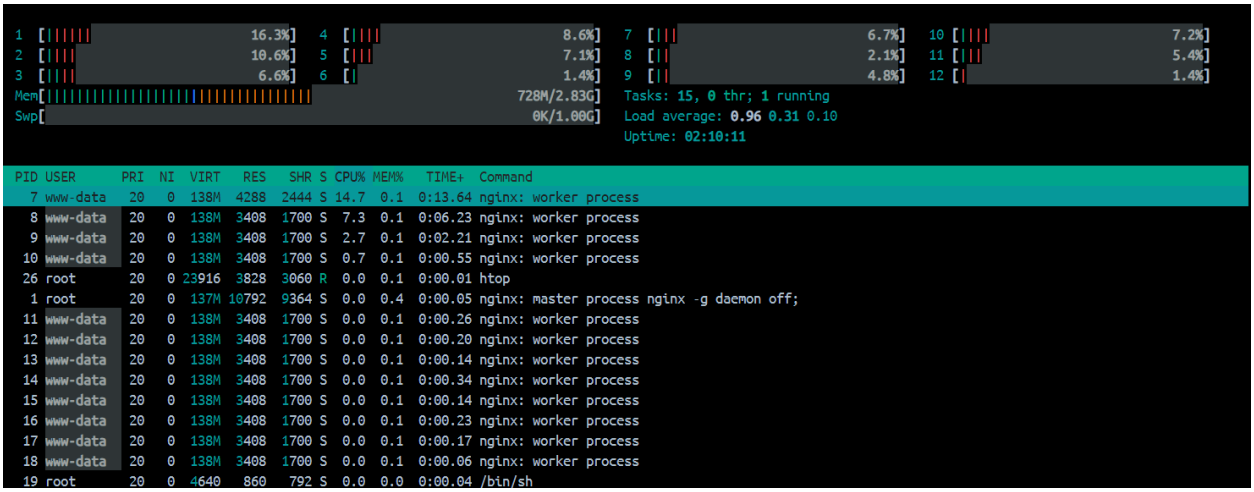


Ultimo minuto:

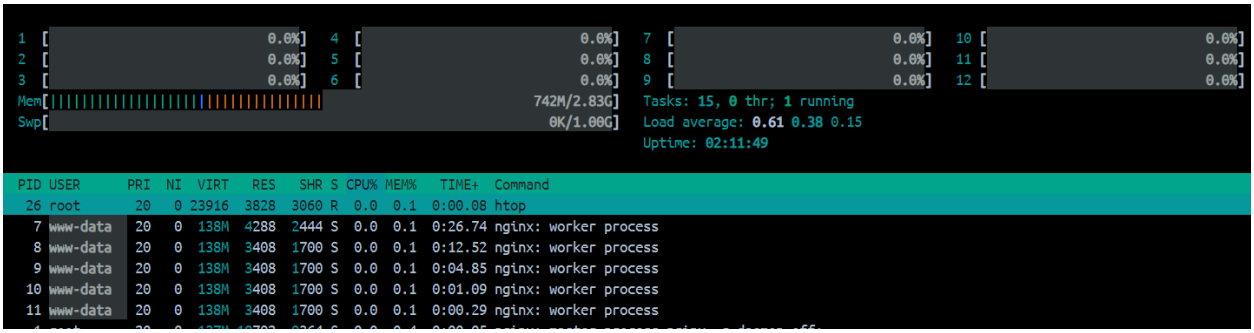


Experimento 2

Observamos el rendimiento:



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.