

Name: Gabriel Oña

Banner: 00320597

Course: Cloud Computing

Homework 1 – Containers and Virtual Machines

Experiment 1

- **Create a Vagrant file to start an Ubuntu VM that hosts a web server with NGINX.**

Link to the Vagrant File - Server:

https://github.com/Gabeinstein/GO-cloud_computing_fall24/blob/main/Server/VM/Vagrantfile

Description:

- Vagrant ubuntu distribution -> hashicorp/bionic64
- Network: public_network
- Port Forwarding -> guest:80, host:8082
- File movement: host: /app, guest: /tmp
- Installation: nginx, htop
- Environment Variable: SERVICE_NAME="Vagrant"
- Envsubst to replace the env variable in the index.html.template and move to
/var/www/html/index.html

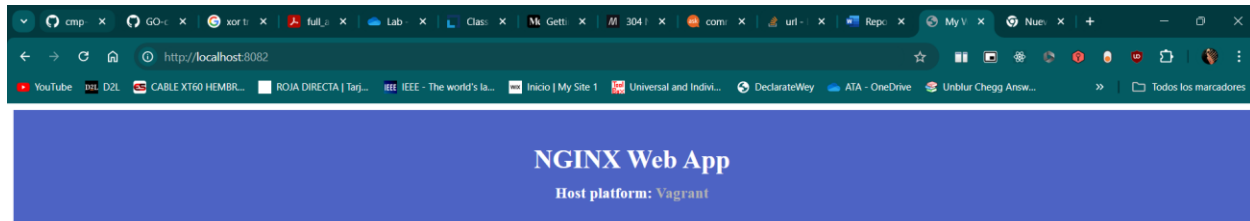
Files in app folder:

- Index.html
- Styles.css

Commands to bring up server:

- `vagrant up`

Web page preview



Class: Cloud Computing 4006 - Fall 24

Name: Gabriel Oña

Experiment 2

- **Create a Docker file to start NGINX container that hosts a web server.**

Link to the Docker File - Server:

https://github.com/Gabeinstein/GO-cloud_computing_fall24/blob/main/Server/Container/Dockerfile

Description:

- Docker image -> `nginx:alpine`
- COPY `/app` to `/usr/share/nginx/html`
- COPY `start.sh` file to use `envsubst`



- Give privileges to start.sh
- Environment Variable: SERVICE_NAME="Vagrant"
- Expose port 80
- Last RUN start.sh

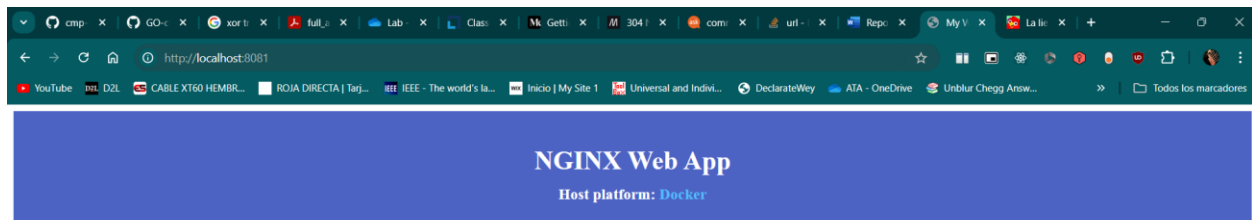
Files in app folder:

- Index.html
- Styles.css

Commands to bring up server:

- Docker buildx build -t my-web-image .
- Docker run -it --rm -d -p 8081:80 --name docker-web-server my-web-image

Web page preview



Class: Cloud Computing 4006 - Fall 24

Name: Gabriel Oña

- **Create a Vagrant File with wrk to overload each server.**

Link to Vagrant File - Client:

https://github.com/Gabeinstein/GO-cloud_computing_fall24/blob/main/Client/VM/Vagrantfile

Description:

- Network: public_network
- Installation: build-essential libssl-dev git zip unzip
- Clone wrk Github repo: git clone https://github.com/wg/wrk.git wrk
- make and copy make file to /usr/local/bin

Commands to bring up client:

- vagrant up
- vagrant ssh

Commands to overload Vagrant server:

- wrk -t14 -c100 -d3m <http://<IP-HOST>:8082>

Replace ip, in my case: 192.168.1.19

For number of threads -t, use the number of cores of your processor

Connections -c, stable connections overloading per thread

- **Vagrant Server Utilization**

1. 14 threads and 100 connections

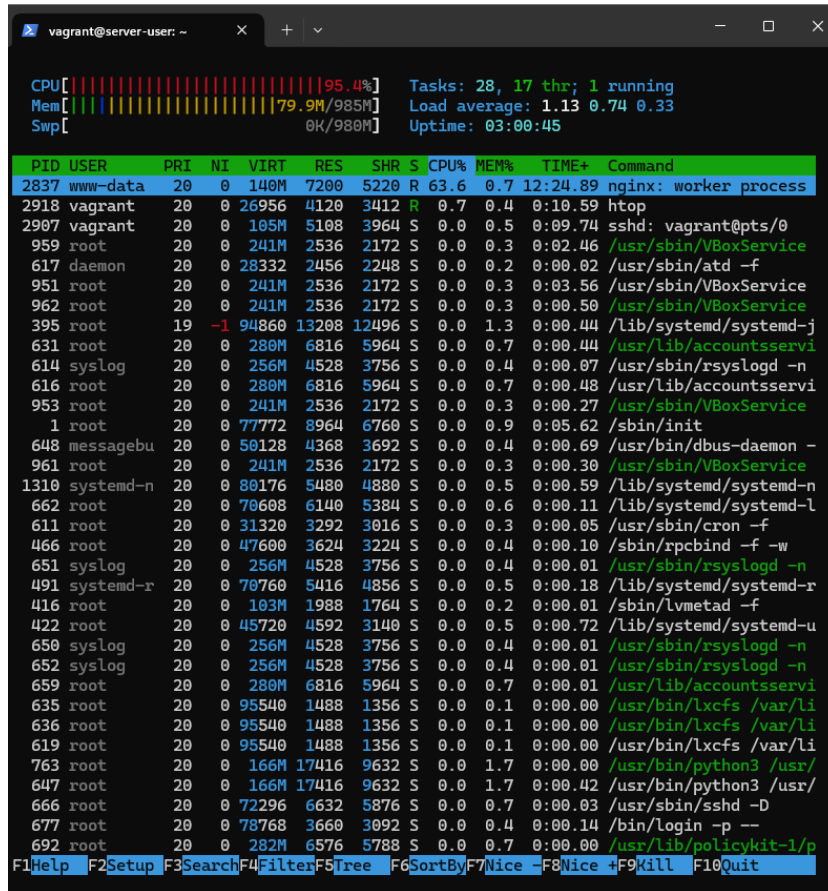


Figure 1. Under Max connections

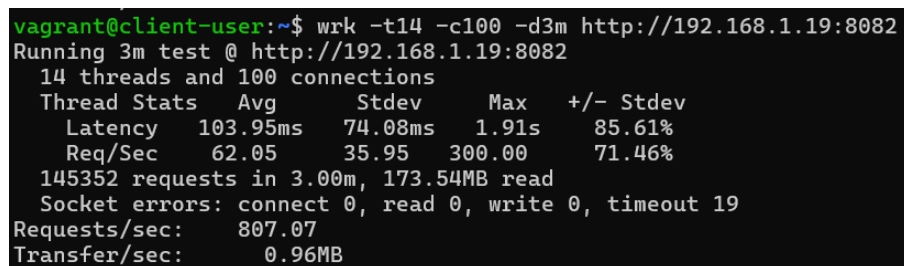


Figure 2. wrk -t14 -c100 report

We can see that CPU% utilization is not 100% and there are few socket errors with the 145352 requests that had happened in 3min.

2. 14 threads and 500 connections

```

vagrant@server-user: ~
CPU[|||||100.0%] Tasks: 28, 17 thr; 1 running
Mem[|||||81.1M/985M] Load average: 0.17 0.43 0.36
Swp[|||||0K/980M] Uptime: 03:07:22

  PID USER      PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
 2837 www-data   20    0  140M  7196  5220  R  67.1  0.7  14:03.43 nginx: worker process
2918 vagrant   20    0  26956  4120  3412  R   0.7  0.4  0:11.57 htop
2907 vagrant   20    0  105M  5108  3964  S   0.0  0.5  0:10.61 sshd: vagrant@pts/0
951  root       20    0  241M  2536  2172  S   0.0  0.3  0:03.70 /usr/sbin/VBoxService
959  root       20    0  241M  2536  2172  S   0.0  0.3  0:02.56 /usr/sbin/VBoxService
953  root       20    0  241M  2536  2172  S   0.0  0.3  0:00.28 /usr/sbin/VBoxService
466  root       20    0  47600  3624  3224  S   0.0  0.4  0:00.11 /sbin/rpcbind -f -w
631  root       20    0  280M  6816  5964  S   0.0  0.7  0:00.45 /usr/lib/accountsservi
1310 systemd-n 20    0  80176  5480  4880  S   0.0  0.5  0:00.62 /lib/systemd/systemd-n
616  root       20    0  280M  6816  5964  S   0.0  0.7  0:00.49 /usr/lib/accountsservi
648 messagebu 20    0  50128  4368  3692  S   0.0  0.4  0:00.72 /usr/bin/dbus-daemon -
961  root       20    0  241M  2536  2172  S   0.0  0.3  0:00.31 /usr/sbin/VBoxService
962  root       20    0  241M  2536  2172  S   0.0  0.3  0:00.51 /usr/sbin/VBoxService
617  daemon     20    0  28332  2456  2248  S   0.0  0.2  0:00.02 /usr/sbin/atd -f
395  root       19   -1  94860 13208 12496  S   0.0  1.3  0:00.44 /lib/systemd/systemd-j
614  syslog     20    0  256M  4528  3756  S   0.0  0.4  0:00.07 /usr/sbin/rsyslogd -n
   1 root       20    0  77772  8964  6760  S   0.0  0.9  0:05.62 /sbin/init
662  root       20    0  70608  6140  5384  S   0.0  0.6  0:00.11 /lib/systemd/systemd-l
611  root       20    0  31320  3292  3016  S   0.0  0.3  0:00.05 /usr/sbin/cron -f
651  syslog     20    0  256M  4528  3756  S   0.0  0.4  0:00.01 /usr/sbin/rsyslogd -n
491  systemd-r 20    0  70760  5416  4856  S   0.0  0.5  0:00.18 /lib/systemd/systemd-r
416  root       20    0  103M  1988  1764  S   0.0  0.2  0:00.01 /sbin/lvmtool -f
422  root       20    0  45720  4592  3140  S   0.0  0.5  0:00.72 /lib/systemd/systemd-u
650  syslog     20    0  256M  4528  3756  S   0.0  0.4  0:00.01 /usr/sbin/rsyslogd -n
652  syslog     20    0  256M  4528  3756  S   0.0  0.4  0:00.01 /usr/sbin/rsyslogd -n
659  root       20    0  280M  6816  5964  S   0.0  0.7  0:00.01 /usr/lib/accountsservi
635  root       20    0  95540 1488  1356  S   0.0  0.1  0:00.00 /usr/bin/lxcfs /var/li
636  root       20    0  95540 1488  1356  S   0.0  0.1  0:00.00 /usr/bin/lxcfs /var/li
619  root       20    0  95540 1488  1356  S   0.0  0.1  0:00.00 /usr/bin/lxcfs /var/li
763  root       20    0  166M 17416  9632  S   0.0  1.7  0:00.00 /usr/bin/python3 /usr/
647  root       20    0  166M 17416  9632  S   0.0  1.7  0:00.42 /usr/bin/python3 /usr/
666  root       20    0  72296  6632  5876  S   0.0  0.7  0:00.03 /usr/sbin/sshd -D
677  root       20    0  78768  3660  3092  S   0.0  0.4  0:00.14 /bin/login -p --
692  root       20    0  282M  6576  5788  S   0.0  0.7  0:00.00 /usr/lib/policykit-1/p
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice F9Kill F10Quit

```

Figure 3. Overloading Vagrant server

```

vagrant@client-user:~$ wrk -t14 -c500 -d3m http://192.168.1.19:8082
Running 3m test @ http://192.168.1.19:8082
14 threads and 500 connections
  Thread Stats   Avg      Stdev     Max   +/-  Stdev
    Latency    346.80ms  167.88ms   1.93s   70.91%
    Req/Sec    75.88    55.96   696.00   72.93%
 179461 requests in 3.00m, 214.27MB read
Socket errors: connect 0, read 45, write 0, timeout 882
Requests/sec: 996.48
Transfer/sec: 1.19MB

```

Figure 4. wrk -t14 -c500 report

As noticed, the number of socket errors has increased significantly. The latency of the server has also increased due to the request overload.

- **Docker Server Utilization**

1. 14 threads and 100 connections

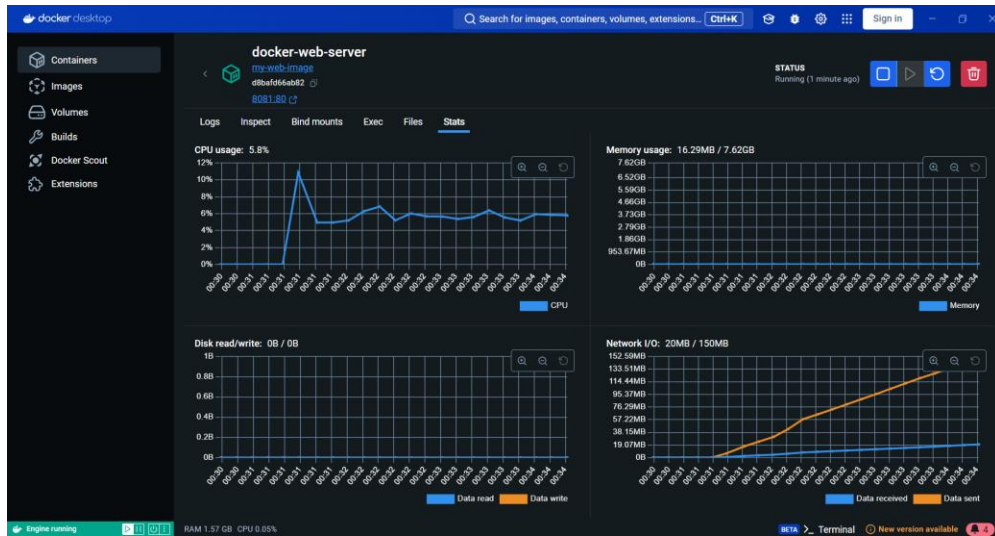


Figure 5. Docker under MAX

```
vagrant@client-user:~$ wrk -t14 -c100 -d3m http://192.168.1.19:8081
Running 3m test @ http://192.168.1.19:8081
14 threads and 100 connections
  Thread Stats   Avg      Stdev     Max   +/-  Stdev
    Latency    100.32ms   56.17ms  420.53ms   70.31%
    Req/Sec    46.76     22.37   228.00    85.06%
  110936 requests in 3.00m, 131.40MB read
Requests/sec:   616.04
Transfer/sec:   747.19KB
```

Figure 6. wrk -t14 -c100 report docker

We can see that the CPU utilization on average was 6% and there is no report of socket errors.

2. 14 threads and 500 connections

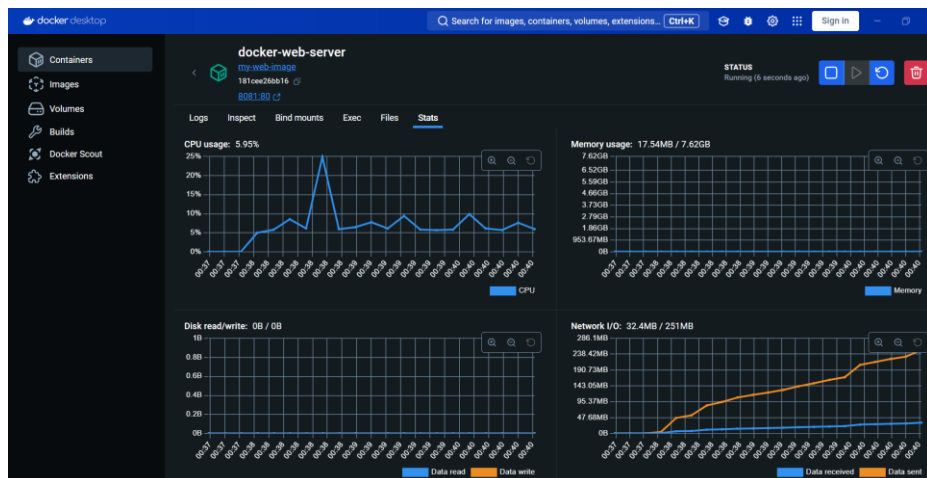


Figure7. Docker stats 500 connections 14 threads

```
vagrant@client-user:~$ wrk -t14 -c500 -d3m http://192.168.1.19:8081
Running 3m test @ http://192.168.1.19:8081
14 threads and 500 connections
Thread Stats      Avg      Stdev     Max    +/-  Stdev
Latency    252.27ms  253.51ms  1.60s   81.59%
Req/Sec    174.54    215.96   1.27k   82.50%
194564 requests in 3.00m, 230.45MB read
Requests/sec: 1079.62
Transfer/sec: 1.28MB
```

Figure 8. wrk -t14 -c500 report docker

We can see that with 500 connections the container is still under overload. CPU% average has increased to 10%. There are no error packets. Unfortunately, the latency has increased as expected due to more connections.

3. 14 threads and 700 connections

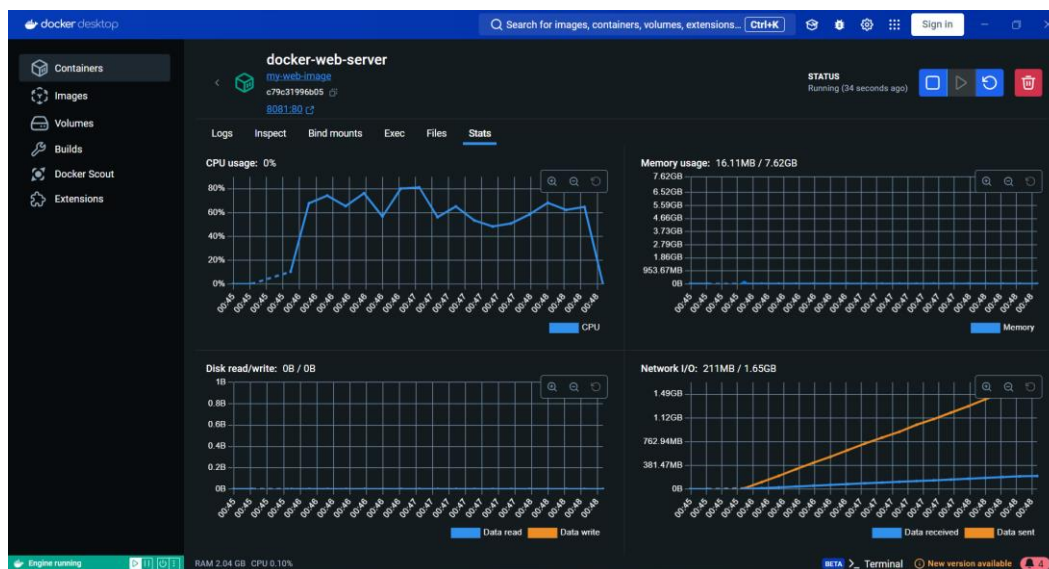


Figure 9. Docker stats 700 connections 14 threads

```
vagrant@client-user:~$ wrk -t14 -c700 -d3m http://192.168.1.19:8081
Running 3m test @ http://192.168.1.19:8081
14 threads and 700 connections
Thread Stats      Avg      Stdev     Max    +/-  Stdev
Latency    124.03ms  139.46ms  1.96s   86.20%
Req/Sec    523.20    204.45   1.57k   69.75%
1198577 requests in 3.00m, 1.39GB read
Socket errors: connect 0, read 0, write 0, timeout 4
Requests/sec: 6655.08
Transfer/sec: 7.88MB
```


Figure 10. wrk -t14 -c700 report docker

With 700 connections per thread, we can see that the CPU% usage has increased to an average of 70%. What this means is that we are achieving the overloading of the server and that is why we are starting to have socket errors.

4. 14 threads and 1000 connections

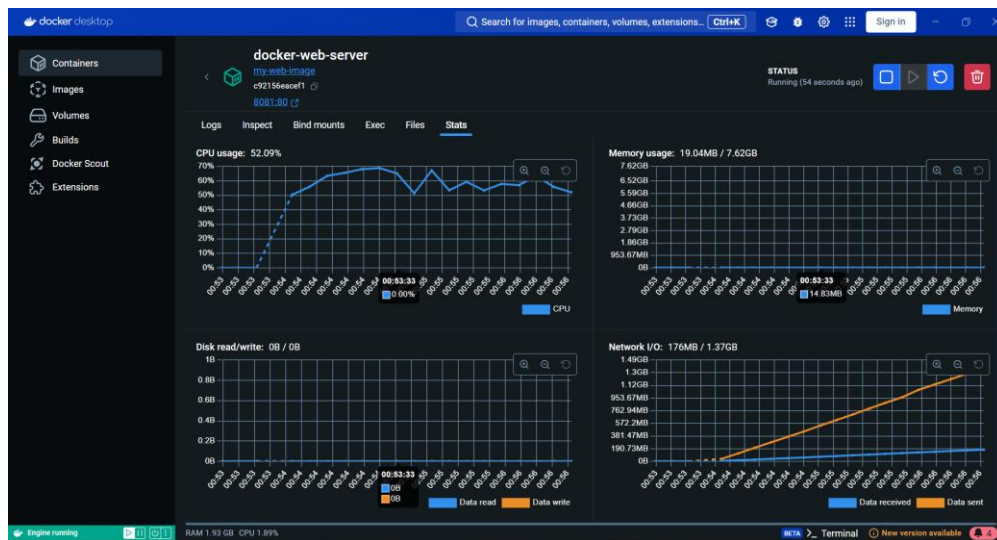


Figure 11. Docker stats 1000 connections 14 threads

```
vagrant@client-user:~$ wrk -t14 -c1000 -d3m http://192.168.1.19:8081
Running 3m test @ http://192.168.1.19:8081
14 threads and 1000 connections
Thread Stats Avg Stdev Max +/- Stdev
Latency 209.98ms 230.82ms 2.00s 86.79%
Req/Sec 442.20 202.81 1.85k 69.06%
998914 requests in 3.00m, 1.16GB read
Socket errors: connect 0, read 0, write 0, timeout 367
Requests/sec: 5546.67
Transfer/sec: 6.57MB
```

Figure 12. wrk -t14 -c1000 report docker

We can observe that the number of socket errors have increased significantly and the latency. These are indicators that the server had been overloaded for the 3 minutes of the experiment.

- **Conclusion**

The implementation using Docker was able to handle more requests per second. This occurred due to the lightweight resources utilization of a container compared with a virtual machine.

To conclude, it is important to know the number of connections that can be handled for each implementation. With this information, we can adequate other resources if we need to allocate more requesters. A good method is using a load balancer to distribute the traffic.