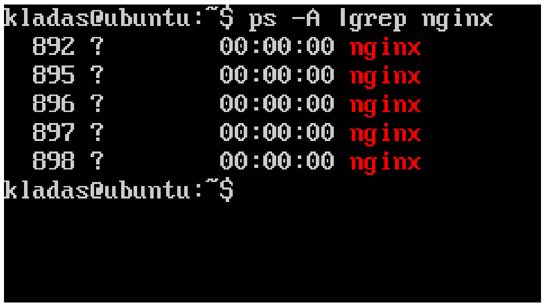
SWAP Practica 3 Report

Christoforos Dellios AZ699940

Konstantinos Ladas AZ700348

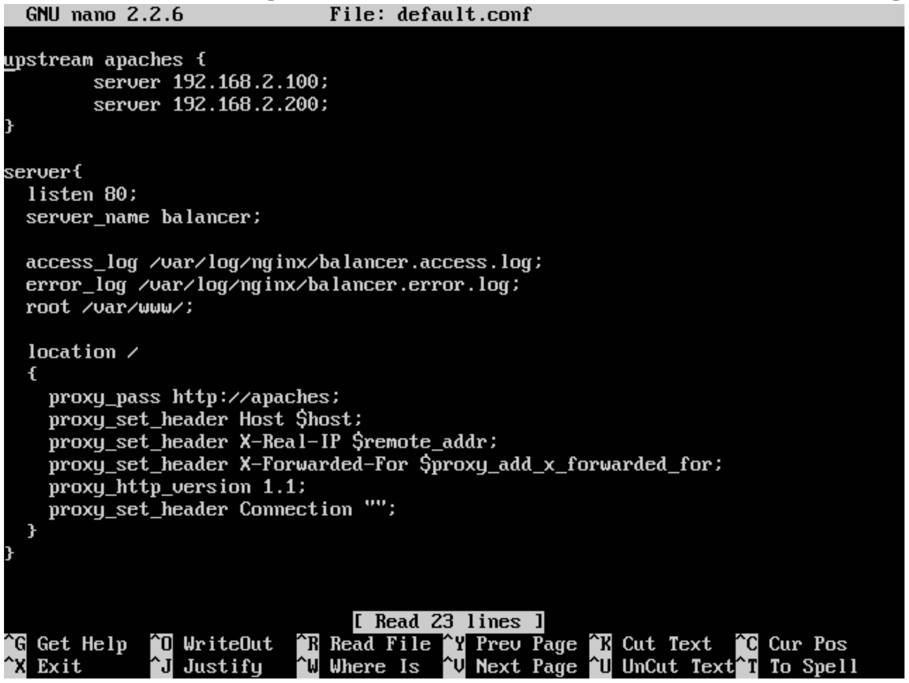
The aim of this practice is to create a balancer and benchmark him along with the two machines of the previous exercises.

First we create a new machine running Ubuntu 12.04 and install nginx there. We already have *apache* running so we stop it using *sudo service apache2 stop* or else nginx will not run.



```
kladas@ubuntu:~$ ps -A |grep nginx
  892 ?        00:00:00 nginx
  895 ?        00:00:00 nginx
  896 ?        00:00:00 nginx
  897 ?        00:00:00 nginx
  898 ?        00:00:00 nginx
kladas@ubuntu:~$
```

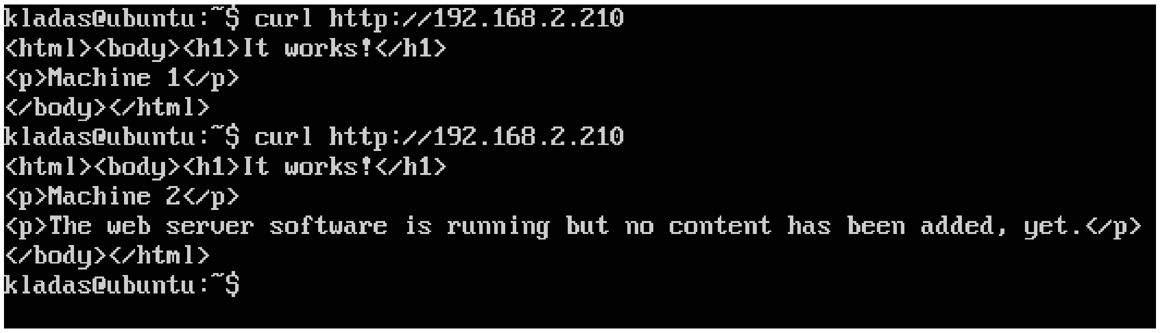
We have to modify the configuration file at /etc/nginx/conf.d/default.conf and use the settings we need. We use the round-robin configuration.



```
  GNU nano 2.2.6             File: default.conf

upstream apaches {
        server 192.168.2.100;
        server 192.168.2.200;
}

server{
  listen 80;
  server_name balancer;

  access_log /var/log/nginx/balancer.access.log;
  error_log /var/log/nginx/balancer.error.log;
  root /var/www/;

  location /
  {
    proxy_pass http://apaches;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_http_version 1.1;
    proxy_set_header Connection "";
  }
}

                              [ Read 23 lines ]
^G Get Help   ^O WriteOut   ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

192.168.2.100 is our first machine and 192.168.2.200 is the second.

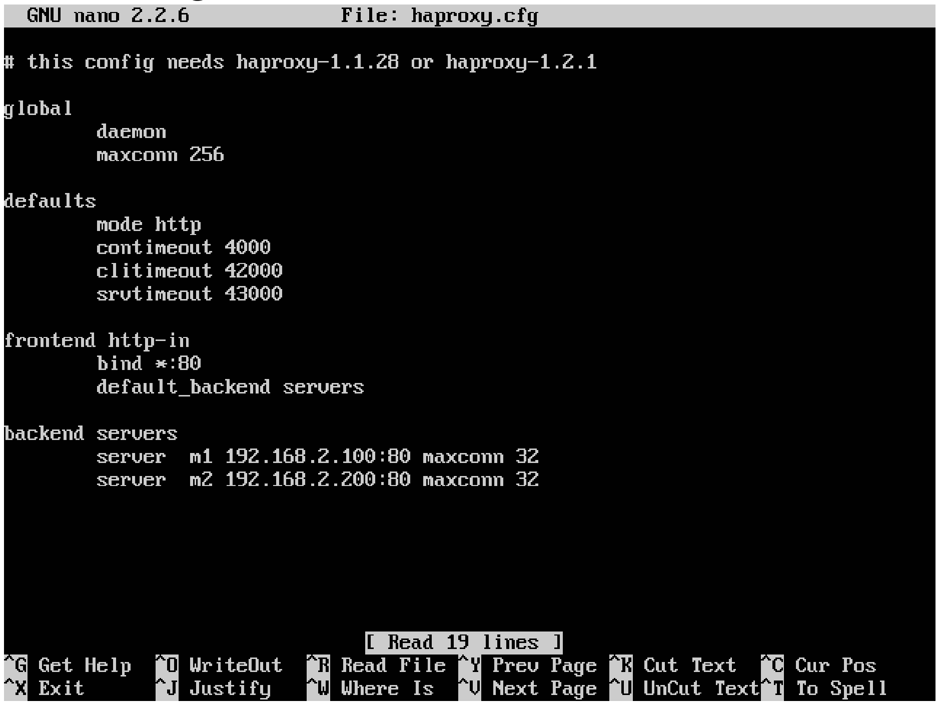
Then we start nginx and run curl to check if the requests to the balancer alternate between machine 1 and machine 2:



```
kladas@ubuntu:~$ curl http://192.168.2.210
<html><body><h1>It works!</h1>
<p>Machine 1</p>
</body></html>
kladas@ubuntu:~$ curl http://192.168.2.210
<html><body><h1>It works!</h1>
<p>Machine 2</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>
kladas@ubuntu:~$
```

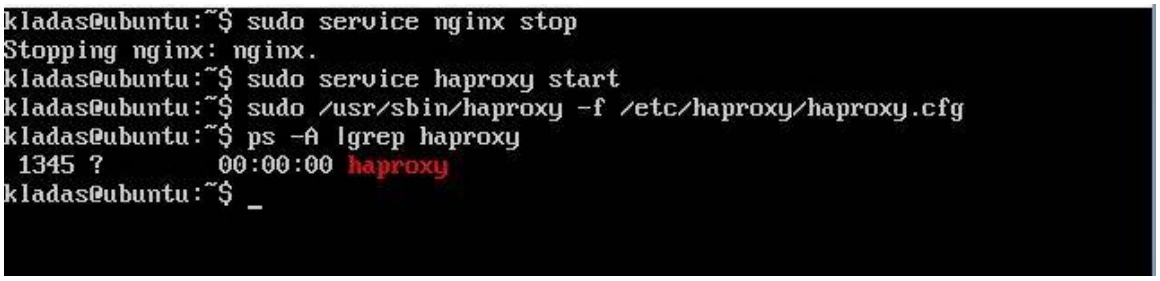
Then we can configure the default.conf file the way we need, adding weights to each server distributing the load how we want. We can add *ip_hash;* which sends the load to one server and if for some reason this server is down, then the load is sent to the another server we have specified. We can also use a number of other configurations like *max_fails*, *fail_timeout*, *down and backup.*

After that, we install *haproxy*. We stop nginx with *sudo service nginx stop* and download haproxy. We modify the file at */ etc / haproxy / haproxy.cf* for the configuration we need:



```
  GNU nano 2.2.6             File: haproxy.cfg

# this config needs haproxy-1.1.28 or haproxy-1.2.1

global
        daemon
        maxconn 256

defaults
        mode http
        contimeout 4000
        clitimeout 42000
        srvtimeout 43000

frontend http-in
        bind *:80
        default_backend servers

backend servers
        server  m1 192.168.2.100:80 maxconn 32
        server  m2 192.168.2.200:80 maxconn 32






                               [ Read 19 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

We run *haproxy* using this command: *sudo / usr / sbin / haproxy -f /etc/haproxy/haproxy.cfg*.

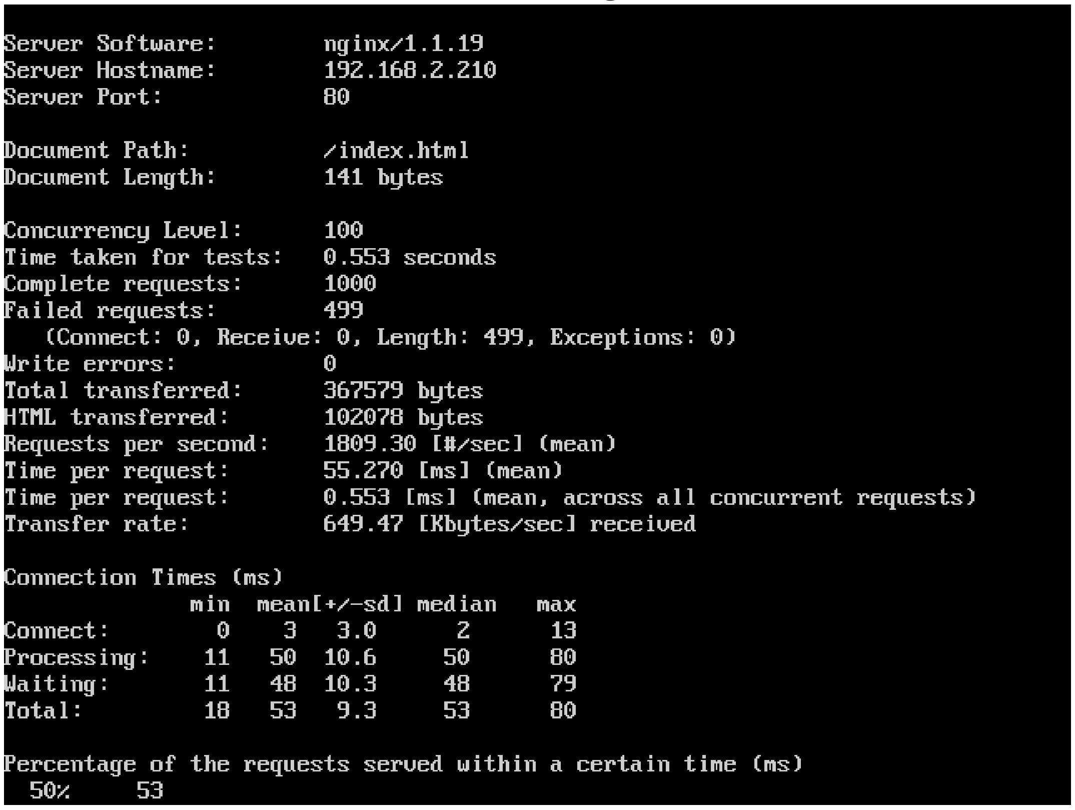


```
kladas@ubuntu:~$ sudo service nginx stop
Stopping nginx: nginx.
kladas@ubuntu:~$ sudo service haproxy start
kladas@ubuntu:~$ sudo /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg
kladas@ubuntu:~$ ps -A |grep haproxy
 1345 ?        00:00:00 haproxy
kladas@ubuntu:~$ _
```

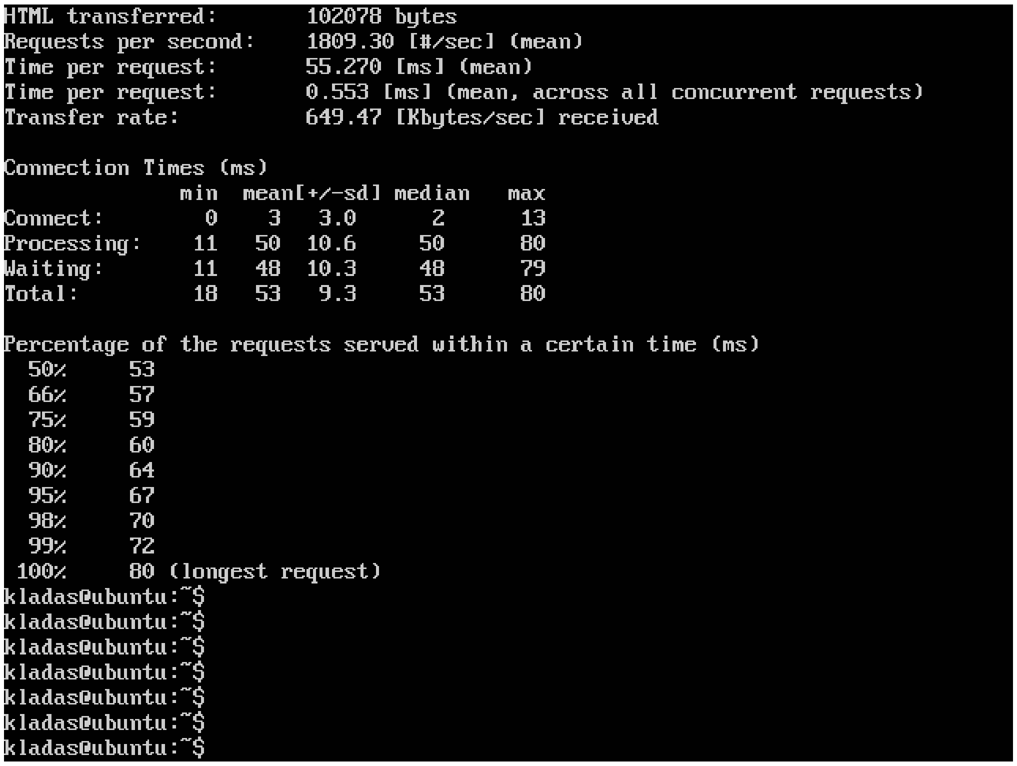
We have set up the nginx and haproxy balancers so we will benchmark them using this apache command from one of the machines (the proper use is from a fourth machine and not from one of the machines that are part of the benchmark process but we use machine 1 instead):

ab -n 1000 -c 10 http://192.168.2.210/index.html

This IP is the balancer's. First, we use nginx and these are the results:

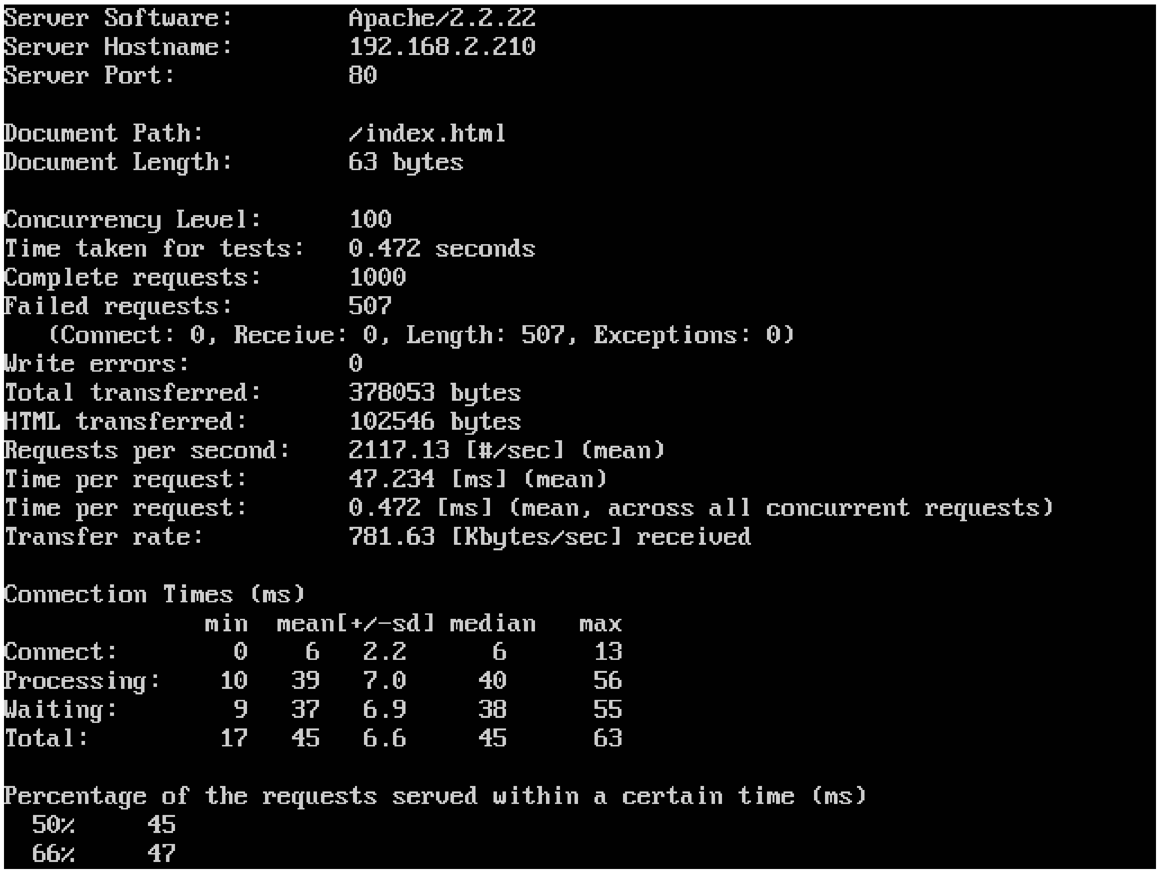


```
Server Software:        nginx/1.1.19
Server Hostname:        192.168.2.210
Server Port:            80

Document Path:          /index.html
Document Length:        141 bytes

Concurrency Level:      100
Time taken for tests:   0.553 seconds
Complete requests:      1000
Failed requests:        499
   (Connect: 0, Receive: 0, Length: 499, Exceptions: 0)
Write errors:           0
Total transferred:      367579 bytes
HTML transferred:       102078 bytes
Requests per second:    1809.30 [#/sec] (mean)
Time per request:       55.270 [ms] (mean)
Time per request:       0.553 [ms] (mean, across all concurrent requests)
Transfer rate:          649.47 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    3   3.0      2      13
Processing:    11   50  10.6     50      80
Waiting:       11   48  10.3     48      79
Total:         18   53   9.3     53      80

Percentage of the requests served within a certain time (ms)
  50%     53
```

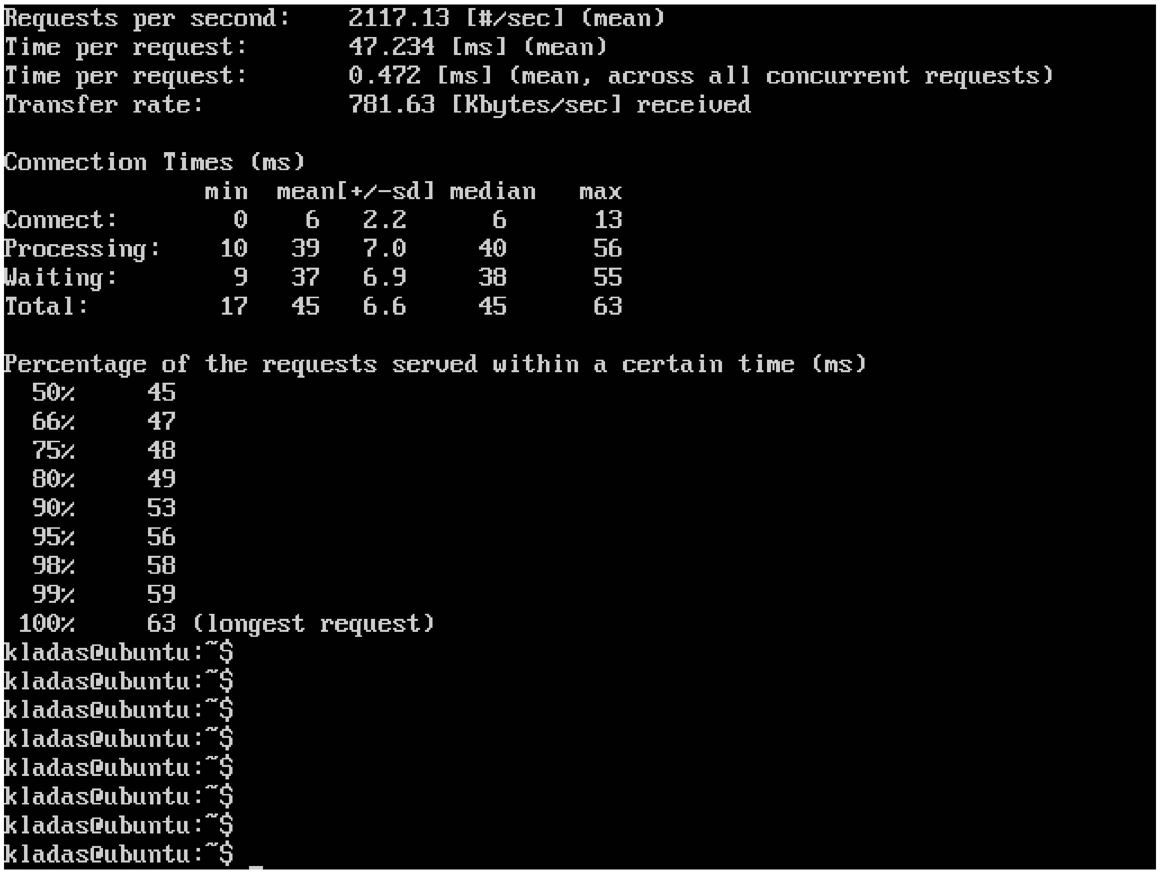


```
HTML transferred:       102078 bytes
Requests per second:    1809.30 [#/sec] (mean)
Time per request:       55.270 [ms] (mean)
Time per request:       0.553 [ms] (mean, across all concurrent requests)
Transfer rate:          649.47 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    3   3.0      2      13
Processing:    11   50  10.6     50      80
Waiting:       11   48  10.3     48      79
Total:         18   53   9.3     53      80

Percentage of the requests served within a certain time (ms)
  50%     53
  66%     57
  75%     59
  80%     60
  90%     64
  95%     67
  98%     70
  99%     72
 100%     80 (longest request)
kladas@ubuntu:~$
kladas@ubuntu:~$
kladas@ubuntu:~$
kladas@ubuntu:~$
kladas@ubuntu:~$
kladas@ubuntu:~$
kladas@ubuntu:~$
```

Then we benchmark using *haproxy:*



```
Server Software:        Apache/2.2.22
Server Hostname:        192.168.2.210
Server Port:            80

Document Path:          /index.html
Document Length:        63 bytes

Concurrency Level:      100
Time taken for tests:   0.472 seconds
Complete requests:      1000
Failed requests:        507
   (Connect: 0, Receive: 0, Length: 507, Exceptions: 0)
Write errors:           0
Total transferred:      378053 bytes
HTML transferred:       102546 bytes
Requests per second:    2117.13 [#/sec] (mean)
Time per request:       47.234 [ms] (mean)
Time per request:       0.472 [ms] (mean, across all concurrent requests)
Transfer rate:          781.63 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    6   2.2      6      13
Processing:    10   39   7.0     40      56
Waiting:        9   37   6.9     38      55
Total:         17   45   6.6     45      63

Percentage of the requests served within a certain time (ms)
  50%     45
  66%     47
```



```
Requests per second:    2117.13 [#/sec] (mean)
Time per request:       47.234 [ms] (mean)
Time per request:       0.472 [ms] (mean, across all concurrent requests)
Transfer rate:          781.63 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    6   2.2      6      13
Processing:    10   39   7.0     40      56
Waiting:        9   37   6.9     38      55
Total:         17   45   6.6     45      63

Percentage of the requests served within a certain time (ms)
  50%     45
  66%     47
  75%     48
  80%     49
  90%     53
  95%     56
  98%     58
  99%     59
 100%     63 (longest request)
kladas@ubuntu:~$
kladas@ubuntu:~$
kladas@ubuntu:~$
kladas@ubuntu:~$
kladas@ubuntu:~$
kladas@ubuntu:~$
kladas@ubuntu:~$
kladas@ubuntu:~$ _
```

We can see that haproxy is a bit faster.