CAT 2

Cloud Computing

Alejandro Pérez Bueno

Nov 26, 2023

Table of Contents

Exercise 1

Basic VM configuration

First and foremost, it is necessary to create the VM and access it with SSH:

```
ssh -i .ssh/id_rsa root@84.88.58.81 -p 55000
```

In order to configure the VM with minimal configuration, one must do the following (as *root*):

```
apt update && apt upgrade -y                                    ①
hostnamectl set-hostname tux                                    ②
echo "127.0.0.1 tux.ccuoc.org tux" >> /etc/hosts                ③
apt install nginx apache2-utils jmeter -y                       ④
```

① Update the system.
② Set hostname to `tux`.
③ Change domain to `ccuog.org`.
④ Install `nginx`, `apache2-utils` and `jmeter`.

Now everything has been set up (see Figure 1).

Set up web service

The website I created is a web version of the first practice I submitted (I write them in `quarto`, a markdown tool that does wonders for these kinds of assignments). It is fairly complex as it has custom images, code formatting, sidebars and other non-textual content.

First, copy the website and its files to the `/var/www/html` directory:

```
cp README.html /var/www/html/index.html                         ①
cp -r README_files/ /var/www/html/                              ②
cp -r img /var/www/html/                                        ③
```

① Main *HTML* file.
② Extra files from quarto to make the website pretty.
③ Images from screenshots I made in the previous CAA.

Then start the `nginx` service:

```
systemctl enable --now nginx
```

At this point our website is up and running! We can check that it is working with `ab` (ApacheBenchmark) and with `JMeter` (see Figure 2).

ℹ Note

Note how in the command from Figure 2a I did not specify the IP of VM, but the domain. This is merely to prove that it is possible to open the website this way too.

You can also simply open a browser on your local machine and enter the IP of the Remote (see Figure 3).

Exercise 2

Basic VM configuration

> ⚠ Warning
>
> The VM from the previous exercise has been deleted, so nothing from the previous exercise will be reused in this one.

Three VMs have to be configured:

1. A machine called `frontend` with a private and a public NIC.
2. A machine called `worker1` with a private NIC.
3. A machine called `worker2` with a private NIC.

On all of them, we have to do the following:

- Update packages and install `apache2`.
- Set hostname and the hosts file appropriately following the same scheme as in the prior exercise.
- Specify the private IPs and domain names for the other VMs in all three machines, as follows:

```
192.168.0.7 frontend
192.168.0.8 worker1
192.168.0.6 worker2
```

You can see how I have set up these VMs in OpenNebula (see @on-vms-ex2).

Frontend setup

Edit the configuration file `/etc/apache2/sites-available/000-default.conf` adding the following lines inside the `VirtualHost` section:

```
<Proxy balancer://mycluster>
  BalancerMember http://worker1:80
  BalancerMember http://worker2:80
  ProxySet lbmethod=byrequests # or ProxySet lbmethod=bytraffic
</Proxy>
ProxyPass /balancer-manager !
ProxyPass / balancer://mycluster/
ProxyPassReverse / http://frontend.ccuoc.org/
ProxyPreserveHost On
<Location /balancer-manager>
  SetHandler balancer-manager
</Location>
```

This configuration sets a balancer for the two websites hosted on the worker VMs with a certain balancer algorithm (installed as a module using `a2enmod`). It also enables the use of the `balancer-manager` extension.

Workers setup

Copy the different websites and their necessary files to each of the worker VMs' `/var/www/html` directory. For this I picked the following:

- For `worker1` VM: Same site from the previous website.
- For `worker2` VM: Similar site from another project.

Testing the results

Once all 3 VMs are up and running, let's make sure that they all three have the `apache` service started and updated to the latest changes made in the configuration files. To do so, we will restart the server:

```
systemctl enable apache2
systemctl restart apache2
```

Then, using `ab` like in the previous exercise we can benchmark the two balancing algorithms we are to compare:

- `byrequests`: distributes traffic evenly to every worker. In this case it would be `50/50`.
- `bytraffic`: distributes the workload based on the load of each worker.

We will compare the two algorithms by running the following command:

```
ab -n 100 -c 10 <FrontendPublicIP>/
```

- `ab`: The same apache utility used in exercise 1.
- `-n 100`: number of petitions made to the server.
- `-c 10`: number of *concurrent* requests made to the server at a time.

As can be seen in the screenshots from Figure 5, `bytraffic` is more efficient at distributing higher workloads, since fewer requests are lost when compared to the `byrequests` algorithm.

You can also verify that the balancer located at `http://<FrontendPublicIP>/balancer-manager` is working (see Figure 6), and that both sites are served in the frontend (see Figure 7).

Exercise 3

> Warning
>
> For the third exercise, I reused some of the previous configurations. Here is what has changed in the setup:
>
> - Recreated `frontend`, without setting up Proxy (I kept running out of space when installing `docker`).
> - Reused `worker1`.
> - Deleted `worker2`.

Basic VM configuration

Let's begin by installing `docker` on both VMs. According to the docker documentation, we must enter the following commands:

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | \
  sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) \
    signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

And then install `docker` and other related utilities:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io \
  docker-buildx-plugin docker-compose-plugin
```

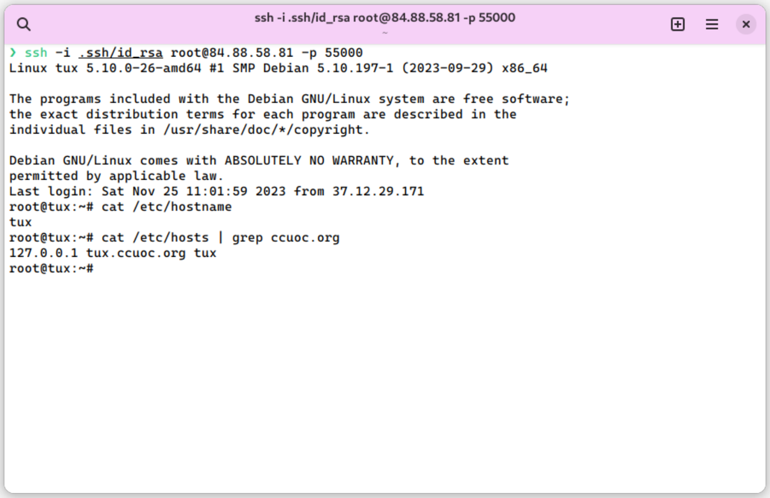
To set up a `swarm` in `docker`, you can do the following on the `frontend` V:

```
docker swarm init --advertise-addr 192.168.0.25
```

This gives you a token that can be used on the `worker` VM as follows:

```
docker swarm join --token <token> 192.168.0.25:2377
```

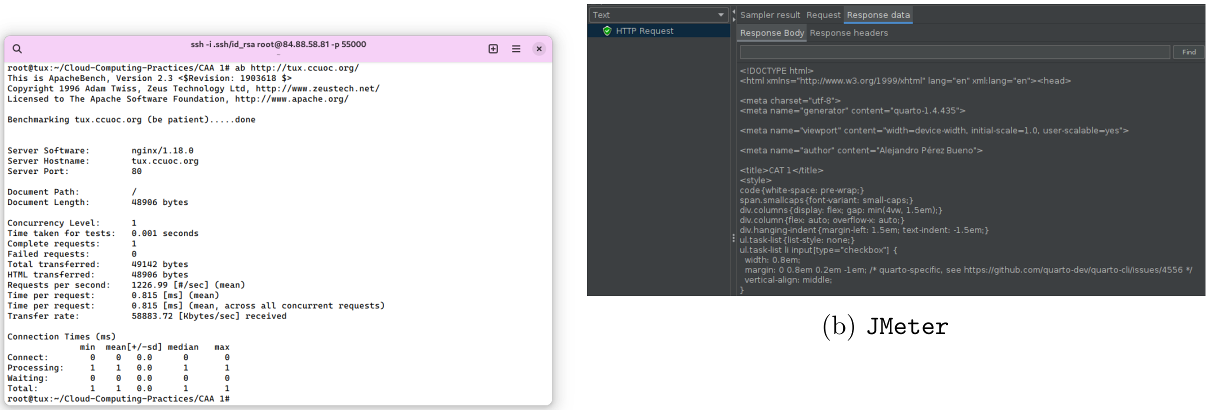
Annex



```
❯ ssh -i .ssh/id_rsa root@84.88.58.81 -p 55000
Linux tux 5.10.0-26-amd64 #1 SMP Debian 5.10.197-1 (2023-09-29) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Nov 25 11:01:59 2023 from 37.12.29.171
root@tux:~# cat /etc/hostname
tux
root@tux:~# cat /etc/hosts | grep ccuoc.org
127.0.0.1 tux.ccuoc.org tux
root@tux:~#
```

Figure 1: Minimal VM Configuration

“‘



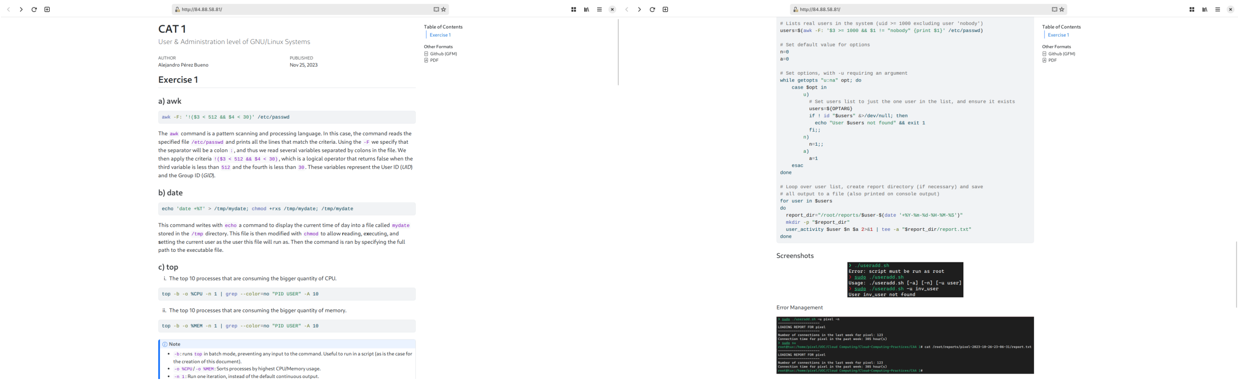
```
root@tux:~/Cloud-Computing-Practices/CAA 1# ab http://tux.ccuoc.org/
This is ApacheBench, Version 2.3 <$Revision: 1903618 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking tux.ccuoc.org (be patient).....done


Server Software:        nginx/1.18.0
Server Hostname:        tux.ccuoc.org
Server Port:            80

Document Path:          /
Document Length:        48906 bytes

Concurrency Level:      1
Time taken for tests:   0.001 seconds
Complete requests:      1
Failed requests:        0
Total transferred:      49142 bytes
HTML transferred:       48906 bytes
Requests per second:    1226.99 [#/sec] (mean)
Time per request:       0.815 [ms] (mean)
Time per request:       0.815 [ms] (mean, across all concurrent requests)
Transfer rate:          58883.72 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:     1    1   0.0      1       1
Waiting:        0    0   0.0      0       0
Total:          1    1   0.0      1       1
root@tux:~/Cloud-Computing-Practices/CAA 1#
```

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en"><head>

<meta charset="utf-8">
<meta name="generator" content="quarto-1.4.435">

<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=yes">

<meta name="author" content="Alejandro Pérez Bueno">

<title>CAT 1</title>
<style>
code{white-space: pre-wrap;}
span.smallcaps{font-variant: small-caps;}
div.columns{display: flex; gap: min(4vw, 1.5em);}
div.column{flex: auto; overflow-x: auto;}
div.hanging-indent{margin-left: 1.5em; text-indent: -1.5em;}
ul.task-list{list-style: none;}
ul.task-list li input[type="checkbox"] {
  width: 0.8em;
  margin: 0 0.8em 0.2em -1em; /* quarto-specific, see https://github.com/quarto-dev/quarto-cli/issues/4556 */
  vertical-align: middle;
}
```

(a) ab

(b) JMeter

Figure 2: Test that the site is running with `ab` and `JMeter`



CAT 1

User & Administration level of GNU/Linux Systems

AUTHOR
Alejandro Pérez Bueno

PUBLISHED
Nov 25, 2023

Exercise 1

a) awk

```
awk -F: '!($3 < 512 && $4 < 30)' /etc/passwd
```

The `awk` command is a pattern scanning and processing language. In this case, the command reads the specified file `/etc/passwd` and prints all the lines that match the criteria. Using the `-F` we specify that the separator will be a colon `:`, and thus we read several variables separated by colons in the file. We then apply the criteria `!($3 < 512 && $4 < 30)`, which is a logical operator that returns false when the third variable is less than `512` and the fourth is less than `30`. These variables represent the User ID (*UID*) and the Group ID (*GID*).

b) date

```
echo 'date +%T' > /tmp/mydate; chmod +rxs /tmp/mydate; /tmp/mydate
```

This command writes with `echo` a command to display the current time of day into a file called `mydate` stored in the `/tmp` directory. This file is then modified with `chmod` to allow reading, executing, and setting the current user as the user this file will run as. Then the command is ran by specifying the full path to the executable file.

c) top

i. The top 10 processes that are consuming the bigger quantity of CPU.

```
top -b -o %CPU -n 1 | grep --color=no "PID USER" -A 10
```

ii. The top 10 processes that are consuming the bigger quantity of memory.

```
top -b -o %MEM -n 1 | grep --color=no "PID USER" -A 10
```

Note

- `-b`: runs `top` in batch mode, preventing any input to the command. Useful to run in a script (as is the case for the creation of this document).
- `-o %CPU` / `-o %MEM`: Sorts processes by highest CPU/Memory usage.
- `-n 1`: Run one iteration, instead of the default continuous output.

```
# Lists real users in the system (uid >= 1000 excluding user 'nobody')
users=$(awk -F: '$3 >= 1000 && $1 != "nobody" {print $1}' /etc/passwd)

# Set default value for options
n=0
a=0

# Set options, with -u requiring an argument
while getopts "u:na" opt; do
    case $opt in
        u)
          # Set users list to just the one user in the list, and ensure it exists
          users=${OPTARG}
          if ! id "$users" &>/dev/null; then
            echo "User $users not found" && exit 1
          fi;;
        n)
          n=1;;
        a)
          a=1
    esac
done

# Loop over user list, create report directory (if necessary) and save
# all output to a file (also printed on console output)
for user in $users
do
  report_dir="/root/reports/$user-$(date '+%Y-%m-%d-%H-%M-%S')"
  mkdir -p "$report_dir"
  user_activity $user $n $a 2>&1 | tee -a "$report_dir/report.txt"
done
```

Screenshots

```
❯ ./useradd.sh
Error: script must be run as root
❯ sudo ./useradd.sh
Usage: ./useradd.sh [-a] [-n] [-u user]
❯ sudo ./useradd.sh -u inv_user
User inv_user not found
```

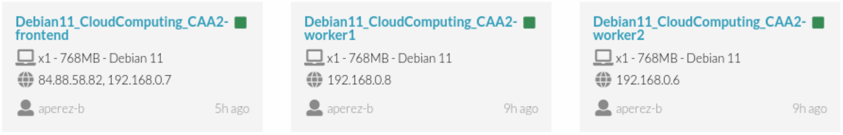
Error Management

```
❯ sudo ./useradd.sh -u pixel -n
--------------------
LOADING REPORT FOR pixel
--------------------
Number of connections in the last week for pixel: 123
Connection time for pixel in the past week: 305 hour(s)
❯ sudo su
root@tux:/home/pixel/UOC/Cloud Computing/Cloud-Computing-Practices/CAA 1# cat /root/reports/pixel-2023-10-26-23-06-31/report.txt
--------------------
LOADING REPORT FOR pixel
--------------------
Number of connections in the last week for pixel: 123
Connection time for pixel in the past week: 305 hour(s)
root@tux:/home/pixel/UOC/Cloud Computing/Cloud-Computing-Practices/CAA 1#
```

(a) Site top

(b) Site image

Figure 3: Test that the site is running with a browser

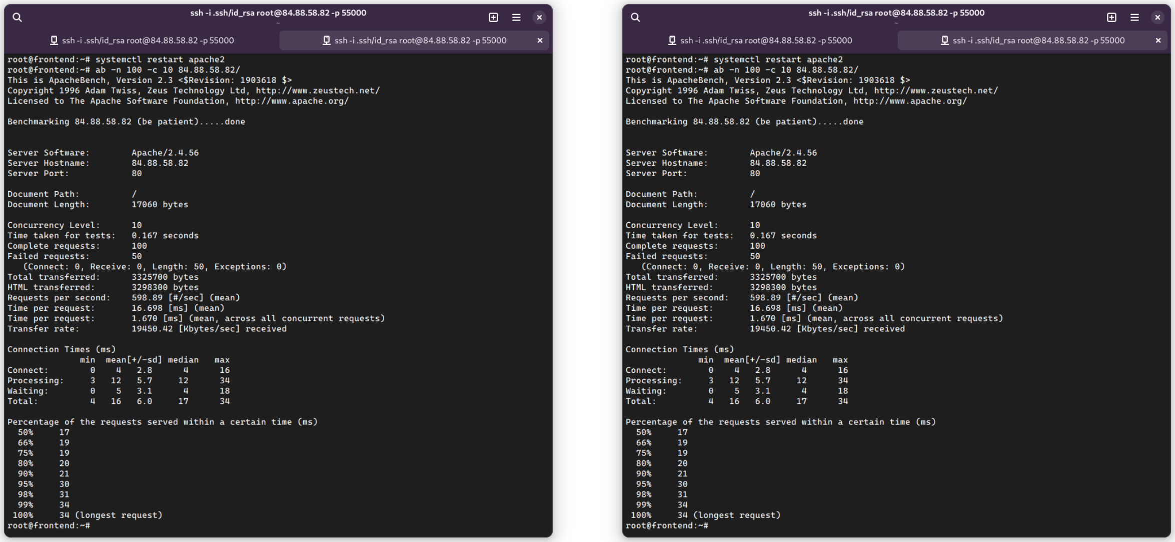


Debian11_CloudComputing_CAA2-frontend 🟩
x1 - 768MB - Debian 11
84.88.58.82, 192.168.0.7
aperez-b 5h ago

Debian11_CloudComputing_CAA2-worker1 🟩
x1 - 768MB - Debian 11
192.168.0.8
aperez-b 9h ago

Debian11_CloudComputing_CAA2-worker2 🟩
x1 - 768MB - Debian 11
192.168.0.6
aperez-b 9h ago

Figure 4: OpenNebula VMs



ssh -i .ssh/id_rsa root@84.88.58.82 -p 55000
ssh -i .ssh/id_rsa root@84.88.58.82 -p 55000
ssh -i .ssh/id_rsa root@84.88.58.82 -p 55000

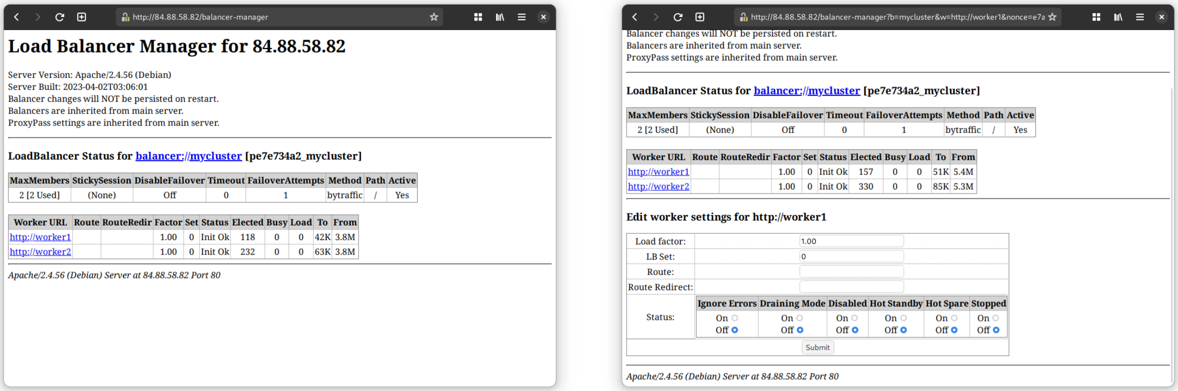
```
root@frontend:~# systemctl restart apache2
root@frontend:~# ab -n 100 -c 10 84.88.58.82/
This is ApacheBench, Version 2.3 <$Revision: 1903618 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 84.88.58.82 (be patient).....done


Server Software:        Apache/2.4.56
Server Hostname:        84.88.58.82
Server Port:            80

Document Path:          /
Document Length:        17060 bytes

Concurrency Level:      10
Time taken for tests:   0.167 seconds
Complete requests:      100
Failed requests:        50
   (Connect: 0, Receive: 0, Length: 50, Exceptions: 0)
Total transferred:      3325700 bytes
HTML transferred:       3298300 bytes
Requests per second:    598.89 [#/sec] (mean)
Time per request:       16.698 [ms] (mean)
Time per request:       1.670 [ms] (mean, across all concurrent requests)
Transfer rate:          19450.42 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    4   2.8      4      16
Processing:     3   12   5.7     12      34
Waiting:        0    5   3.1      4      18
Total:          4   16   6.0     17      34

Percentage of the requests served within a certain time (ms)
  50%     17
  66%     19
  75%     19
  80%     20
  90%     21
  95%     30
  98%     31
  99%     34
 100%     34 (longest request)
root@frontend:~#
```

(a) `lbmethod=byrequests` - 50 failed requests

```
root@frontend:~# systemctl restart apache2
root@frontend:~# ab -n 100 -c 10 84.88.58.82/
This is ApacheBench, Version 2.3 <$Revision: 1903618 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 84.88.58.82 (be patient).....done


Server Software:        Apache/2.4.56
Server Hostname:        84.88.58.82
Server Port:            80

Document Path:          /
Document Length:        17060 bytes

Concurrency Level:      10
Time taken for tests:   0.167 seconds
Complete requests:      100
Failed requests:        50
   (Connect: 0, Receive: 0, Length: 50, Exceptions: 0)
Total transferred:      3325700 bytes
HTML transferred:       3298300 bytes
Requests per second:    598.89 [#/sec] (mean)
Time per request:       16.698 [ms] (mean)
Time per request:       1.670 [ms] (mean, across all concurrent requests)
Transfer rate:          19450.42 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    4   2.8      4      16
Processing:     3   12   5.7     12      34
Waiting:        0    5   3.1      4      18
Total:          4   16   6.0     17      34

Percentage of the requests served within a certain time (ms)
  50%     17
  66%     19
  75%     19
  80%     20
  90%     21
  95%     30
  98%     31
  99%     34
 100%     34 (longest request)
root@frontend:~#
```

(b) `lbmethod=bytraffic` - 28 failed requests

Figure 5: Test the command `ab -n 100 -c 10 84.88.58.82` with two balancing algorithms



Load Balancer Manager for 84.88.58.82

Server Version: Apache/2.4.56 (Debian)
Server Built: 2023-04-02T03:06:01
Balancer changes will NOT be persisted on restart.
Balancers are inherited from main server.
ProxyPass settings are inherited from main server.

LoadBalancer Status for balancer://mycluster [pe7e734a2_mycluster]

MaxMembers	StickySession	DisableFailover	Timeout	FailoverAttempts	Method	Path	Active
2 [2 Used]	(None)	Off	0	1	bytraffic	/	Yes

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	Busy	Load	To	From
http://worker1			1.00	0	Init Ok	118	0	0	42K	3.8M
http://worker2			1.00	0	Init Ok	232	0	0	63K	3.8M

Apache/2.4.56 (Debian) Server at 84.88.58.82 Port 80

(a) Main `balancer-manager` page

http://84.88.58.82/balancer-manager?b=mycluster&w=http://worker1&nonce=e7a

Balancer changes will NOT be persisted on restart.
Balancers are inherited from main server.
ProxyPass settings are inherited from main server.

LoadBalancer Status for balancer://mycluster [pe7e734a2_mycluster]

MaxMembers	StickySession	DisableFailover	Timeout	FailoverAttempts	Method	Path	Active
2 [2 Used]	(None)	Off	0	1	bytraffic	/	Yes

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	Busy	Load	To	From
http://worker1			1.00	0	Init Ok	157	0	0	51K	5.4M
http://worker2			1.00	0	Init Ok	330	0	0	85K	5.3M

Edit worker settings for http://worker1

Load factor:	1.00
LB Set:	0
Route:	
Route Redirect:	

Status:

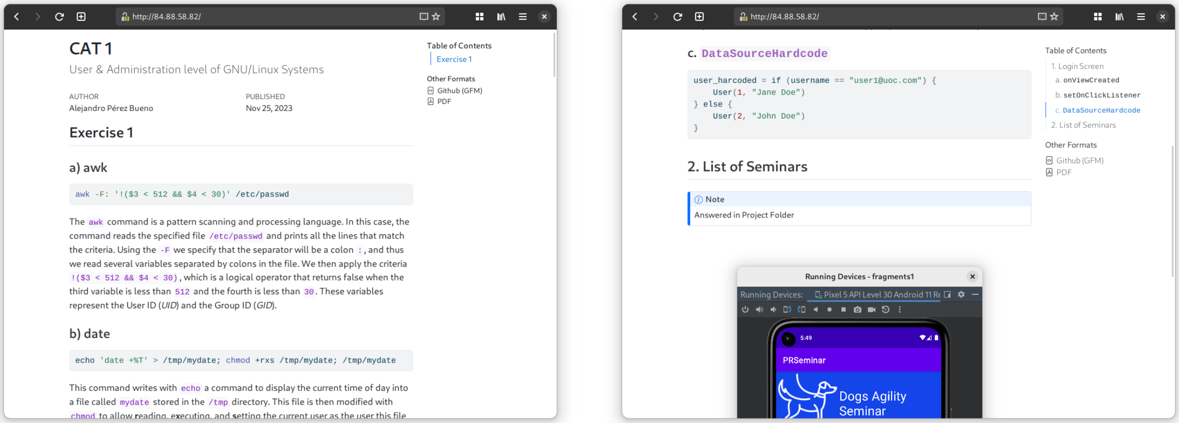
Ignore Errors	Draining Mode	Disabled	Hot Standby	Hot Spare	Stopped
On ○ Off ●	On ○ Off ●	On ○ Off ●	On ○ Off ●	On ○ Off ●	On ○ Off ●

Submit

Apache/2.4.56 (Debian) Server at 84.88.58.82 Port 80

(b) Config for `worker1` in `balancer-manager`

Figure 6: `balance-manager` page



http://84.88.58.82/

CAT 1

User & Administration level of GNU/Linux Systems

AUTHOR
Alejandro Pérez Bueno

PUBLISHED
Nov 25, 2023

Exercise 1

a) awk

```
awk -F: '!($3 < 512 && $4 < 30)' /etc/passwd
```

The `awk` command is a pattern scanning and processing language. In this case, the command reads the specified file `/etc/passwd` and prints all the lines that match the criteria. Using the `-F` we specify that the separator will be a colon `:`, and thus we read several variables separated by colons in the file. We then apply the criteria `!($3 < 512 && $4 < 30)`, which is a logical operator that returns false when the third variable is less than `512` and the fourth is less than `30`. These variables represent the User ID (*UID*) and the Group ID (*GID*).

b) date

```
echo 'date +%T' > /tmp/mydate; chmod +rxs /tmp/mydate; /tmp/mydate
```

This command writes with `echo` a command to display the current time of day into a file called `mydate` stored in the `/tmp` directory. This file is then modified with `chmod` to allow reading, executing, and setting the current user as the user this file

Table of Contents
Exercise 1

Other Formats
Github (GFM)
PDF

http://84.88.58.82/

c. `DataSourceHardcode`

```
user_harcoded = if (username == "user1@uoc.com") {
    User(1, "Jane Doe")
} else {
    User(2, "John Doe")
}
```

2. List of Seminars

Note
Answered in Project Folder

Table of Contents
1. Login Screen
a. `onViewCreated`
b. `setOnClickListener`
c. `DataSourceHardcode`
2. List of Seminars

Other Formats
Github (GFM)
PDF

Running Devices - fragments1
Running Devices:
Pixel 5 API Level 30 Android 11 Re
5:49
PRSeminar
Dogs Agility Seminar

(a) Page from `worker1` accessed from `frontend`

(b) Page from `worker2` accessed from `frontend`

Figure 7: Pages served by each worker balanced from the `frontend`