CAT 3

Cloud Computing

Alejandro Pérez Bueno

Jan 03, 2024

Table of Contents

Exercise 1

Basic VM configuration

First and foremost, it is necessary to create the EC2 instance that will host our web service, named `caa3-ami-1` with the tag `aperez-b`.

See Figure 1 for more details.

We can then connect to it via ssh from our machine with the private key we downloaded in the creation screen, as follows:

```
chmod 400 ~/.ssh/keypair-aperez-b.pem
ssh -i ~/.ssh/keypair-aperez-b.pem ec2-user@ec2-3-94-211-15.compute-1.amazonaws.com
```

> **Note**
>
> In order to properly connect to the remote instance, I had to change the default routing table to allow connection with the internet.

To configure the VM with minimal configuration, one must do the following:

```
sudo yum update -y                                          (1)
sudo yum upgrade -y
sudo hostnamectl set-hostname aperez-b-master               (2)
sudo yum install httpd -y                                   (3)
sudo systemctl enable --now httpd
```

(1) Update the system.
(2) Set hostname to `aperez-b-master`.
(3) Install and set up `httpd`.

Now everything has been set up.

Set up web service

> **Note**
>
> I used the HTML equivalent to the submission I worked on for CAA 2

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 took in the previous CAA.

At this point our website is up and running at http://ec2-3-94-211-15.compute-1.amazonaws.com. We can check that it is working with `JMeter` (see Figure 2a).

You can also simply open a browser on your local machine and enter the IP of the Remote (see Figure 2b).

Exercise 2

Basic VM configuration

> **Warning**
>
> The VM from the previous exercise will NOT 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 `httpd`.
- Set hostname following the same scheme as in the prior exercise.
- Worker VMs will be called `aperez-b-worker1` and `aperez-b-worker2`.

Here is an example of the code ran on `worker1`:

```
sudo hostnamectl set-hostname aperez-b-worker1
sudo yum update -y
sudo yum upgrade -y
sudo yum install httpd -y
sudo systemctl enable --now httpd
```

Frontend setup

Create the configuration file `/etc/httpd/conf.d/balancer.conf` adding the following lines:

```
<VirtualHost *:80>
    ServerName ec2-18-210-13-85.compute-1.amazonaws.com
    ProxyPreserveHost On
    <Proxy balancer://mycluster>
        BalancerMember http://172.31.6.227:80
        BalancerMember http://172.31.6.197:80
        ProxySet lbmethod=byrequests # or ProxySet lbmethod=bytraffic
    </Proxy>
    <Location /balancer-manager>
        SetHandler balancer-manager
    </Location>
    ProxyPass /balancer-manager !
    ProxyPass / balancer://mycluster/
    ProxyPassReverse / balancer://mycluster/
```

```
</VirtualHost>
```

This configuration sets a balancer for the two websites hosted on the worker VMs with a certain balancer algorithm and 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 exercise.
- 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 `httpd` service started and updated to the latest changes made in the configuration files. To do so, we will restart the server:

```
systemctl enable httpd
systemctl restart httpd
```

See Figure 6 to verify that the websites are served properly from the `frontend`.

Then, using `ab` 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 3, `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 4), and that both sites are served in the frontend (see Figure 5).

Exercise 3

AWS VM configuration

For this last part I merely followed the 5-module tutorial from the post *Deploy WordPress with Amazon RDS* carefully to properly install and configure `WordPress`.

See Figure 7 to check the settings of EC2 and RDS after following the aforementioned tutorial.

WordPress installation

See Figure 8 to see how the WordPress site looked before and after setting up the database and the Wordpress app itself.

> **Note**
>
> At first, I could not upload anything to the media section of the WordPress app. Turns out I was lacking permissions to write in the specific directory. To fix it, I ran the following command from ssh:
>
> ```
> sudo chown -R apache:apache /var/www/html/
> ```

WordPress setup

After configuring the database and the VM, it is time to log in with a new WordPress user. In my case I created a new user `aperez-b` with my `UOC` email address and started adding blog posts to the site. I added a total of six entries, with various types of content and media. Some have lists, images, links, quotes, video, audio, and so on (see Figure 10).

> **Note**
>
> As it is hard to show the WordPress site with just screenshots, I recorded a short video showcasing every post (thus proving that it all works as expected)

WordPress performance

See Figure 9 to view the result of the test performed by PageSpeed Insights.

> **Warning**
>
> In Figure 9, the url appears to be different from all other screenshots from this exercise. This is because I had to enable the site to be accessed by anyone (at first I had set it up so that only my IP could access the website), and as a result a new Public IP address was assigned

Conclusion

In this CAA I have worked with AWS for the first time, and I cannot say it was a piece of cake. Nevertheless, it was a fun journey.The overall experience of working with it to deploy applications has been a valuable learning opportunity. By understanding the technologies for the deployment of cloud computing infrastructures securely and efficiently, I was able to deploy a balanced proxy using Apache2 with a front-end and two workers, as well as a full-fledged WordPress server. Overall, the experience has provided me with a solid foundation in cloud computing that I can build upon in future projects.

Annexes

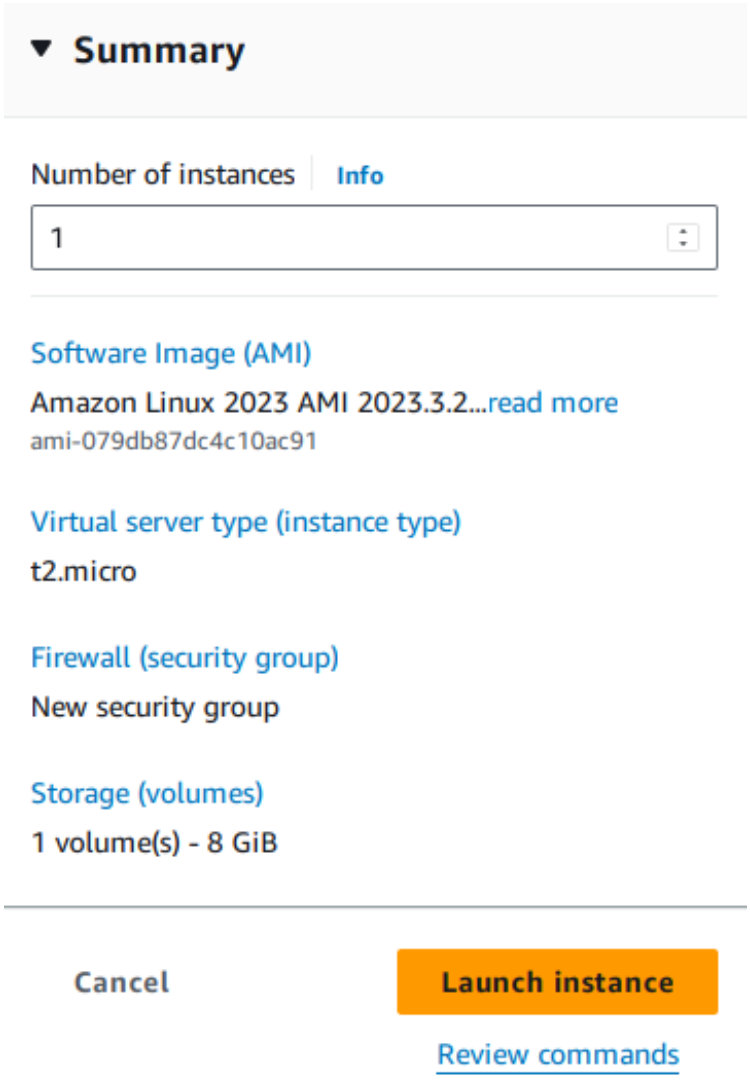
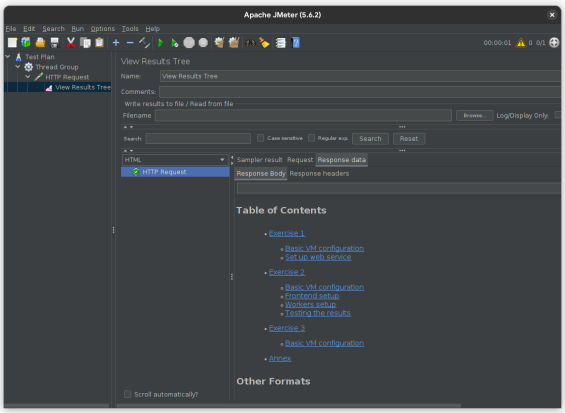
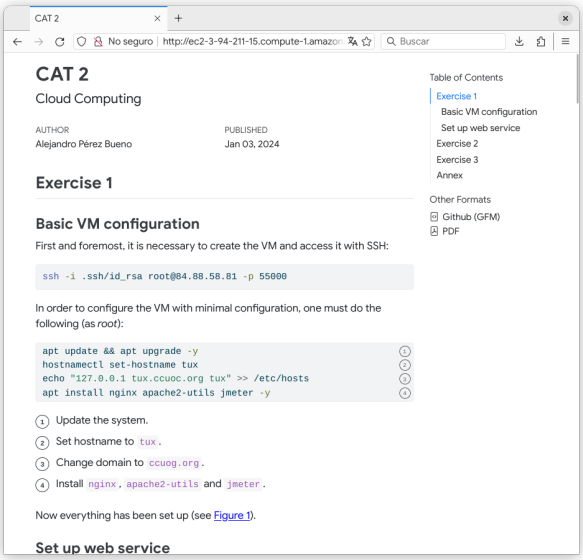


Figure 1: Add instance preview



(a) JMeter plan result



(b) Web server up and running

Figure 2: Website running on browser and in JMeter

```
❯ ab -n 100 -c 10 http://ec2-18-210-13-85.compute-1.amazonaws.com/
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 ec2-18-210-13-85.compute-1.amazonaws.com (be patient).....done


Server Software:        Apache/2.4.58
Server Hostname:        ec2-18-210-13-85.compute-1.amazonaws.com
Server Port:            80

Document Path:          /
Document Length:        42284 bytes

Concurrency Level:      10
Time taken for tests:   5.516 seconds
Complete requests:      100
Failed requests:        50
   (Connect: 0, Receive: 0, Length: 50, Exceptions: 0)
Total transferred:      2994400 bytes
HTML transferred:       2967200 bytes
Requests per second:    18.13 [#/sec] (mean)
Time per request:       551.579 [ms] (mean)
Time per request:       55.158 [ms] (mean, across all concurrent requests)
Transfer rate:          530.15 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:      120  161  25.5    154     220
Processing:   252  338  88.6    302     616
Waiting:      124  160  27.3    148     212
Total:        382  499 106.8    456     821

Percentage of the requests served within a certain time (ms)
  50%    456
  66%    480
  75%    524
  80%    582
  90%    621
  95%    777
  98%    821
  99%    821
 100%    821 (longest request)

^ ~/UOC/Cloud Computing/Cloud-Computing-Practices/CAA 3 main ?1
❯
```

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

```
❯ ab -n 100 -c 10 http://ec2-18-210-13-85.compute-1.amazonaws.com/
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 ec2-18-210-13-85.compute-1.amazonaws.com (be patient).....done


Server Software:        Apache/2.4.58
Server Hostname:        ec2-18-210-13-85.compute-1.amazonaws.com
Server Port:            80

Document Path:          /
Document Length:        17060 bytes

Concurrency Level:      10
Time taken for tests:   5.448 seconds
Complete requests:      100
Failed requests:        30
   (Connect: 0, Receive: 0, Length: 30, Exceptions: 0)
Total transferred:      2489920 bytes
HTML transferred:       2462720 bytes
Requests per second:    18.36 [#/sec] (mean)
Time per request:       544.780 [ms] (mean)
Time per request:       54.478 [ms] (mean, across all concurrent requests)
Transfer rate:          446.34 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:      123  183 175.8    147    1170
Processing:   251  307  79.4    293    1001
Waiting:      124  158  46.8    151     575
Total:        380  490 192.1    446    1477

Percentage of the requests served within a certain time (ms)
  50%    446
  66%    458
  75%    465
  80%    471
  90%    576
  95%    673
  98%   1461
  99%   1477
 100%   1477 (longest request)

^ ~/UOC/Cloud Computing/Cloud-Computing-Practices/CAA 3 main ?1
❯
```

(b) `lbmethod=bytraffic` - 30 failed requests

Figure 3: Test the command `ab -n 100 -c 10 <FrontendPublicIP>` with two balancing algorithms

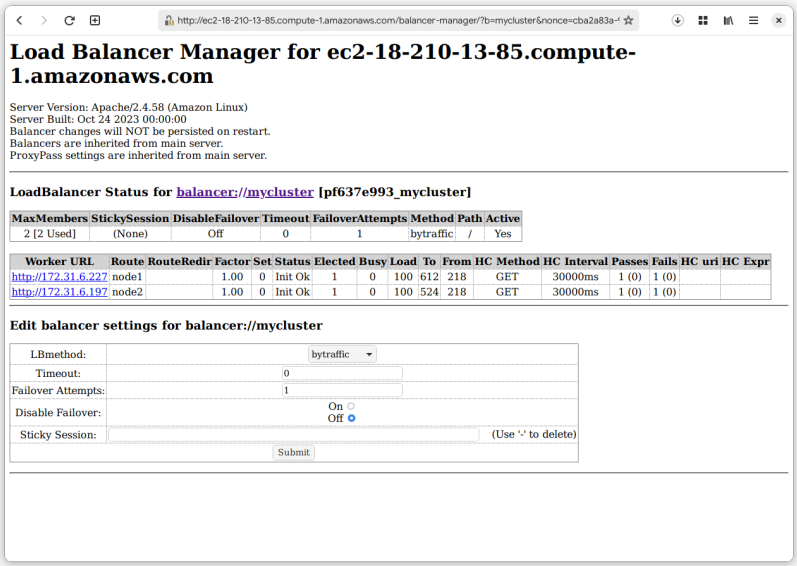
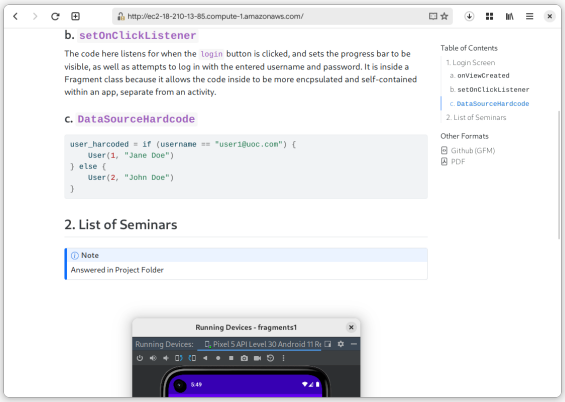
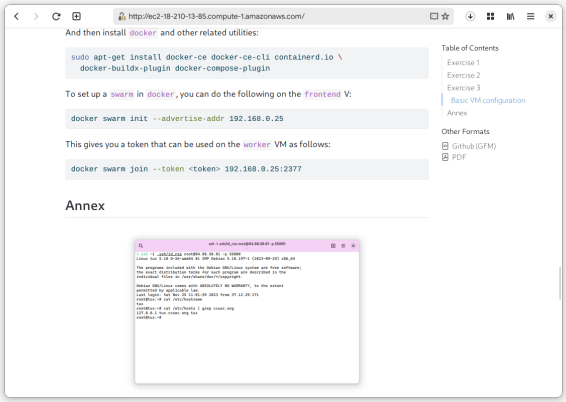


Figure 4: `balancer-manager` working



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



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

Figure 5: Pages served by each worker balanced from the `frontend`

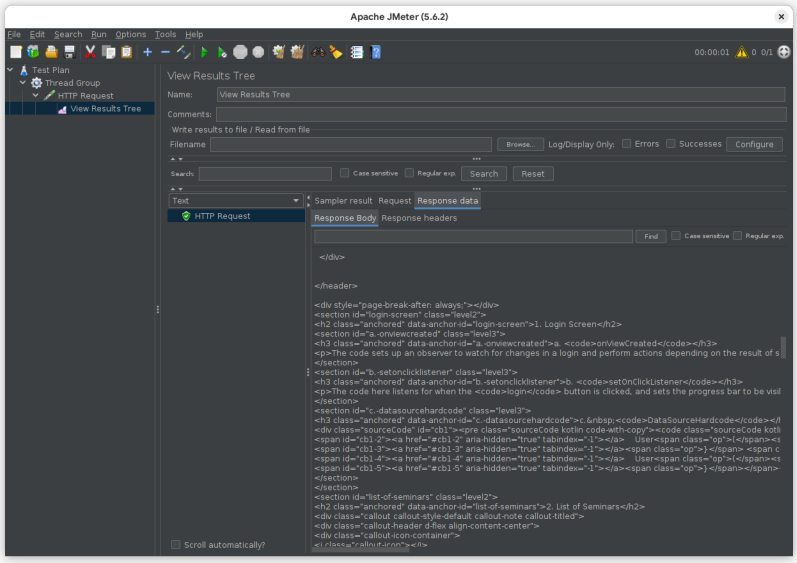
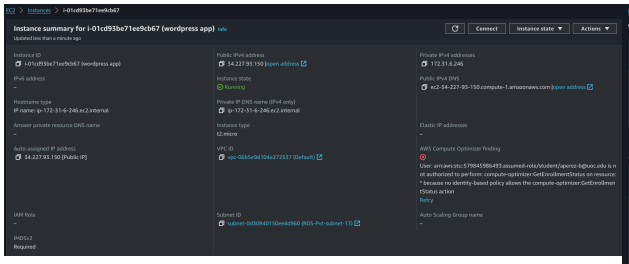
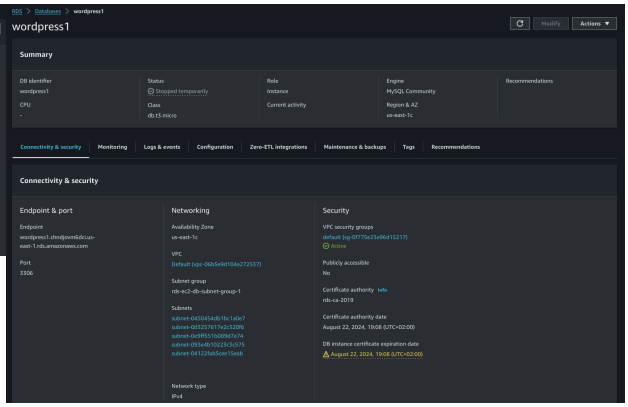


Figure 6: JMeter plan result

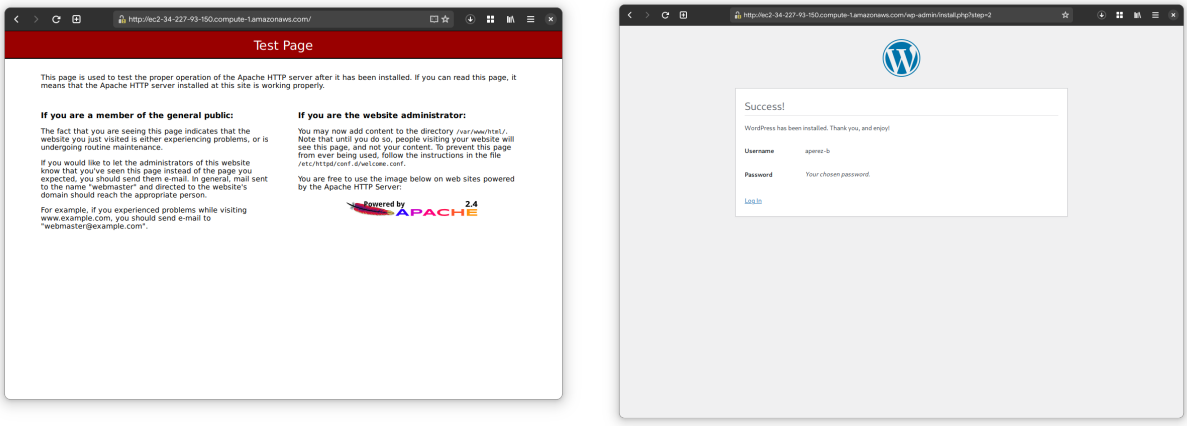


(a) EC2 settings



(b) RDS settings

Figure 7: EC2 and RDS configuration



(a) pre-setup

(b) post-setup

Figure 8: WordPress pre-setup and post-setup

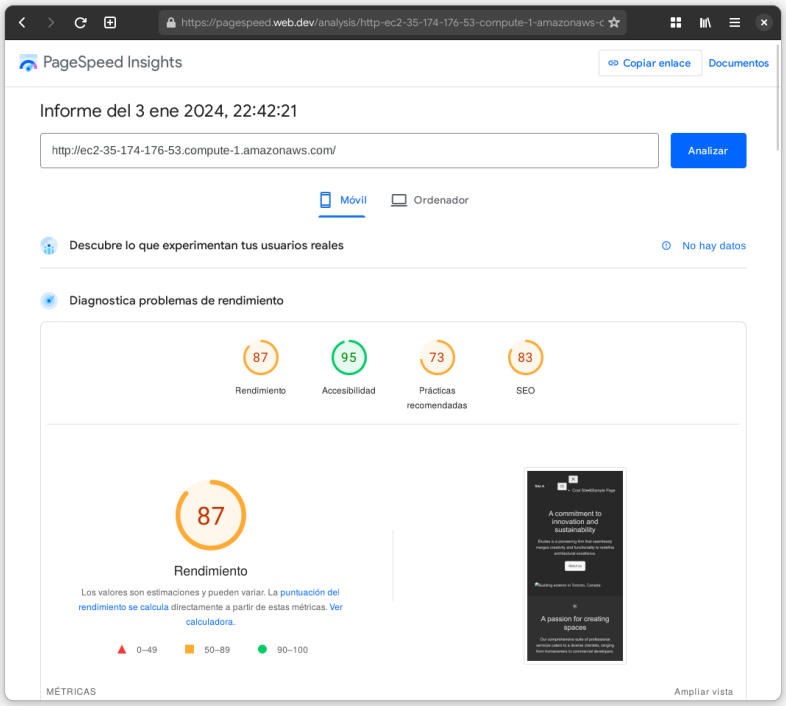


Figure 9: PageSpeed Insights report

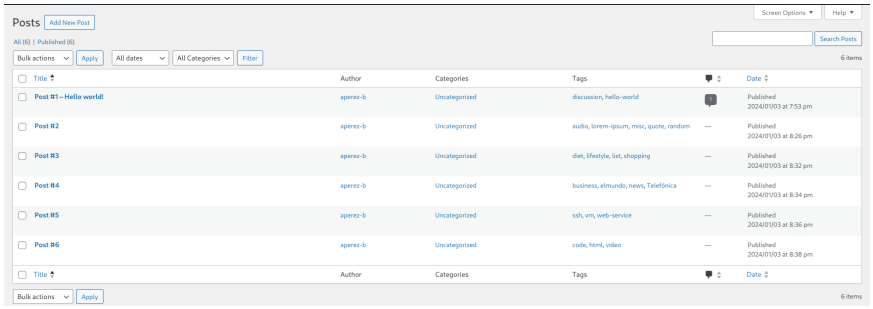


Figure 10: WordPress post overview