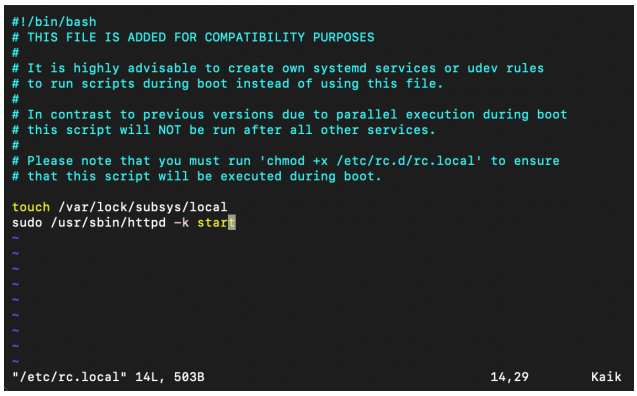
Cloud Computing Task 3

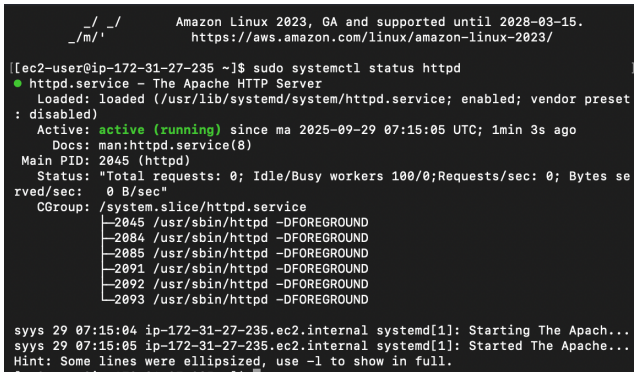
I started an instance with the Amazon Linux 2 AMI and chose the t3.small (since the recommended t2.small is not supported on this AMI). I made sure SSH and HTTP are both allowed and connected to it to update the packages next.

All packages were up to date so next I downloaded the Apache server and PHP.

I opened the file with vi with "`sudo vi /etc/rc.local`" and added the line.



I then saved it with :wq and rebooted to make sure it works



Which it does.

Next I created the index.php at /var/www/html and pasted the moodle code into it.

I then made sure it has the right permissions with "`sudo chmod 644 index.php`"

I opened a firefox tab and inserted http://3.80.50.206/index.php, and was greeted with the Hello! My name is... so it seems to be working.

Then I created an AMI using the instance, with which I will next create two more instances. Node1 and Node2 were both created with the AMI and t3.small.
Now I go to the Load Balancers section of EC2 and create one.

While making the load balancer I noticed I needed to make a target group, in which I updated the health check path to /index.php. Then I added the original instance, and the two node ones to it. For security groups I noticed I had a different one for each instance so I added all

three to the load balancer.

I had some trouble getting it going since my target group did not get connected to the load balancer. Next I sat a couple hours thinking about why my instances all had the health status unused and why my load balancer says 503 service temporarily unavailable.

This was solved by me checking every possible setting until I realized that the load balancer didn't have the same AZ as the instances so I deleted it and made a new one, which of course instantly fixed my problem.

Next I downloaded httperf and got to recording the graph while testing.

While testing I realized I had not created the necessary auto scaling group so the test just spiked all three instances at ones, which was not the plan. So I created one which I connected to my target group with CPU utilization set to the target value of 60.

Okay so after a few runs with different command I could not get a CPU spike over 10% and only had one instance running because of it.

I tried the commands:
httperf --server=ALBTest-155975176.us-east-1.elb.amazonaws.com --uri=/index.php --wsess=500,20,1 --rate=5 --timeout=5

httperf --server=ALBTest-155975176.us-east-1.elb.amazonaws.com --uri=/index.php --wsess=300,10,1 --rate=2 --timeout=5

httperf --server=ALBTest-155975176.us-east-1.elb.amazonaws.com --uri=/index.php --wsess=600,8,2 --rate=1 --timeout=5

But none of these got me anywhere, the first had a 10% spike, the second was 5% and third only 3%, so something was definitely not working as intended.

So due to another frustrating problem I decided to just start again from a blank sheet, without doing steps 8 and 9 and instead focusing on making a launch template and an auto scaling group. This approach started the same by creating an instance, downloading the php and httpd and making the file that contained the php code from moodle.

After this I used this model to create an AMI for the launch template. I used t2.small for it as was recommended, since it works in the template, but not when creating an instance with the

linux version (weird). I used the same security group as with the instance and added #!/bin/bash sudo systemctl start httpd to the advanced details to make sure apache starts without having to ssh in.

I then created the application load balancer in which I made sure that the same AZ was included as the instance, and two others as extras. I updated the health check path and then it was on to the auto scaling group. I chose my template and chose the same AZ's as before. I connected it to my target group for the load balancer part and made sure the minimum capacity was 1 and maximum 3. I changed the CPU utilization limit to 50% since I didn't even get near 60 last time.
Then it was time for testing, and I used the command in the pdf file to try and strain my setup. It quickly rose to 48% and from there to 90, where it sat for many minutes. This was enough for the load balancer to launch another instance within 5 minutes of the load being over 50%. When the other instance launched it got the load down to 50%, and then to 30% and then the test ended which made it drop to close to 0. After about 20 minutes of waiting I finally saw that the number of instances was back to 1, which meant that the load balancer worked as planned, it just had long delays just in case.

Video link: https://drive.google.com/file/d/113vKwXRu01m291nGZGkLw-2Wsoo0aJkN/view?usp=sharing

As can be seen the cpu utilization drops to close to zero at 17:15 but the other instance closes at around 17:32. The load balancer did its supposed job perfectly by launching an instance after the CPU utilization was high with only one instance, and by killing the extra instance when the utilization was low. It also always kept one instance, which was one of the requirements.

I have read a bit about auto load balancers when I wrote my bachelors, but making one myself was of course a completely new experience. I found it surprisingly straight forward and really automated once all the correct steps were taken.

The challenging part was of course getting it to work, since the workflow clearly needed to stray away from the given steps in the pdf file. But once I had all the different parts I needed, I really found the task interesting and fun. The testing took longer than expected but I was very satisfied once I got everything working.