Cloud Computing

Assignment 3

You can collect up to 20 points for this assignment

Elastic web services: resources auto-scaling

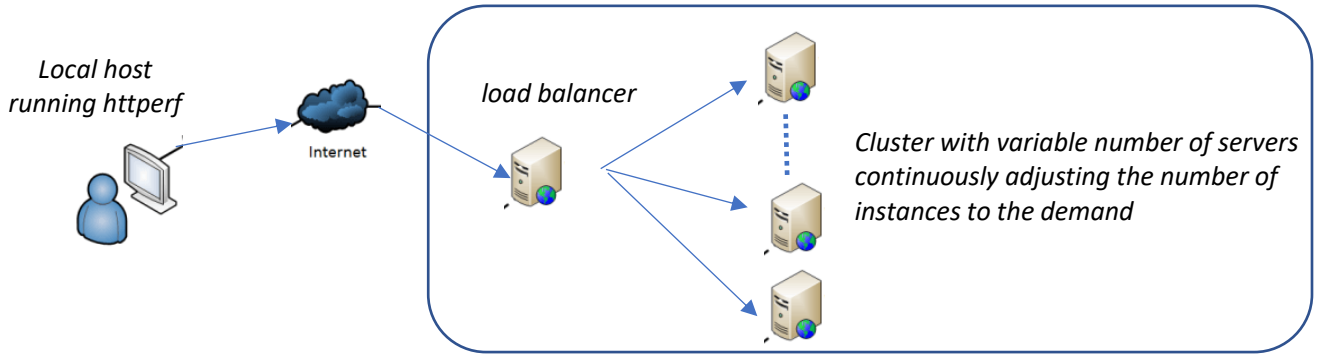
- **Do start working on this assignment several days before the deadline!**

This is an individual assignment. *Discussing the assignment tasks and specific issues with other course participants is allowed and even encouraged. However, you should be the only author of all the solutions you provide in this assignment. Teamwork, pair programming, or copying solutions or program code from other persons is consider plagiarism and it will be handled following the Åbo Akademi University protocol for such cases.*

Instructions:

- Upload your assignment as a single **PDF** file through the Moodle system!
- The file should be named **Assignment3_LastName_Firstname_studentID.pdf**.
- **Each page** of your submission should have:
  - Your name
  - Student ID at Åbo Akademi University
  - A page number / total number of pages
- **Pay attention to the readability of your report!**

The goal of this assignment is to implement a web service able to automatically scale resources according to the load. The resource capacity used to provide the service should constantly adapt to the current load generated by service requests. The number of instances should provide an acceptable service performance level. The number of instances need to be scaled down when the demand decreases in order to keep the costs down.



Requirements:

- You must create a load balancer distributing requests to a variable number of instances. You are free to choose your instance type.

  Note: AWS Academy student accounts support only following instance types: nano, micro, small, medium, and large.

- In theory, the maximum number of instances will be defined by your account limitations (in the EC2 dashboard click on the link “Limits” from the left panel to see your limits). But it looks like that with a student account, the maximum size of an auto-scaling group is 3 instances.

- You must have at minimum one instance running at any time.

- The instances behind the load balancer must run the **index.php** file available on Moodle. This file will simulate a task taking some variable time to execute and will load the CPUs of the instances with a variable load.
  https://moodle.abo.fi/mod/resource/view.php?id=928669

- The instructions for installing and running PHP and Apache are given at the end of this PDF file.

- Your auto-scaling policy should update the number of instances by adding or removing one instance at a time (using “*Scale the Auto Scaling group using step or simple scaling policies”*)

- Make sure your load balancer and target group are attached to your Auto Scaling group

- To generate requests on the load balancer (and instances), you can use from **your local machine** the tool **httperf**: https://linux.die.net/man/1/httperf

  For example, the following command:

  ```
  httperf --server ec2-52-59-56-175.eu-central-1.compute.amazonaws.com --uri=/index.php --wsess=600,5,2 --rate 1 --timeout 5
  ```

  Will generate a total of 600 sessions at a rate of 1 session every second. Each session consisting of 5 calls that are spaced out by 2 seconds. Requests are targeting the **index.php** file hosted on *ec2-52-59-56-175.eu-central-1.compute.amazonaws.com*

  Depending on the type of instance you are using, you will need to play with the number of calls per session and the delay between each call. For example, the load generated by one request on a ***t2.small*** instance will range from around **3 to 10% CPU load** over a period from around **100ms to 350ms**.

  Also, feel free to execute several *httperf* programs in parallel with different time intervals. You will need to find good enough values for the number of calls per session and the delay to not too quickly load your instances, but at the same time be able to load them enough to trigger several times the need to add instances. Start with relatively small values for the

number of generated http requests. An excessively large number of requests might trigger firewall rules set by your ISP.

For this assignment, you need to get familiar with:

- Create an Auto Scaling group using the Amazon EC2 launch wizard: https://docs.aws.amazon.com/autoscaling/ec2/userguide/create-asg-ec2-wizard.html
- Step and simple scaling policies for Amazon EC2 Auto Scaling: https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-scaling-simple-step.html
- Auto Scaling Groups: http://docs.aws.amazon.com/autoscaling/latest/userguide/AutoScalingGroup.html
- Launch Configurations: http://docs.aws.amazon.com/autoscaling/latest/userguide/LaunchConfiguration.html
- Scaling Plans: http://docs.aws.amazon.com/autoscaling/latest/userguide/scaling_plan.html
- Amazon CloudWatch: https://aws.amazon.com/cloudwatch/
- Httperf: https://linux.die.net/man/1/httperf

Report

You should prepare a report documenting the work performed during this exercise. The report should contain information on *what you did*, *how you did it* and *why you did it* this way.

You report should contain the following:

- (4p) A short description of the steps you took to implement your elastic web service, and the used parameters
- (4p) The used *httperf* command(s) to create load on the instances
- (4p) Screenshots and link(s) to online video(s)***** showing the CPU utilization across the instances over adequate time windows. You should annotate the graph(s) and/or videos with the corresponding events on their timelines (like start of *httperf* execution(s), instance creations, instance terminations, etc.).
- (8p) A detailed analysis on the obtained behavior and reactivity of your web service to load variations (increases and decreases of load). With the obtained behavior and reaction time, discuss some possible (concrete) web applications that could benefit from your auto-scaling mechanism for cloud resources.

At the end of the report, you should also provide a reflection on what you learned during this exercise. This section could provide answers to the following questions:

- Have you learned anything completely new?
- Did anything surprise you?
- Did you find anything challenging? Why?
- Did you find anything satisfying? Why?

Remember to terminate all VMs, terminate the load balancer, deregister your AMI and delete its corresponding Snapshot, when you are done!

* You are free to use any video online service (e.g., YouTube, Google Drive) to host your video(s)

Instructions for installing and running PHP and Apache

Step 1: Create a new instance (for example a t2.small) using the **Amazon Linux 2 AMI** (instead of the default Amazon Linux 2023 AMI)

- Open at least the port 80 (http) and 22 (ssh)

Step 2: Update the packages of the Linux distribution

Step 3: Install Apache server on your VM

- *sudo yum install httpd*

Step 4: Install PHP

- *sudo yum install php*

Step 5: Make sure Apache starts on boot

- edit the file /etc/rc.local
- add the following line at the end of the file: *sudo /usr/sbin/httpd -k start*
    - If you have no clue how to do it, you can use the vi editor https://www.howtogeek.com/102468/a-beginners-guide-to-editing-text-files-with-vi
    - Feel free to install your preferred text editor

Step 6: Create a file **index.php** in /var/www/html containing the code in the following file (be sure you have write access to the folder).
https://moodle.abo.fi/mod/resource/view.php?id=928669

Step 7: Start Apache (httpd) and test it can serve your index.php file (i.e. open a browser on your local computer, and type the Public DNS of your VM)

- *sudo /usr/sbin/httpd -k start*

Step 8: If needed, create and save an AMI out of your instance running Apache (it will take around 2-3 minutes before the newly created image becomes available).

This step might not be needed when using an Auto Scaling group.
Note: it is recommended to reboot the instance before creating the image.

Step 9: When your AMI is available, create two new instances instantiated with your newly created AMI.

- Make sure they all have port 80 open
- You might want to give a unique name to the machine via the console (for example, node1, node2, etc…)

This step might not be needed when using an Auto Scaling group.

Step 10: Create a load balancer via the AWS console

- Create an **Application Load Balancer**
- Make sure it serves port 80, and its port 80 is open
- Update the ping path for the health check
- Add all instances running Apache to the load balancer
- Tags are optional

Step 11: Test your load balancer by pointing a browser on your local machine to the URL of the load balancer. Refresh several times the page.