Cloud Computing

Simone Salvatore La Milia

December 15, 2023

1 Second Activity

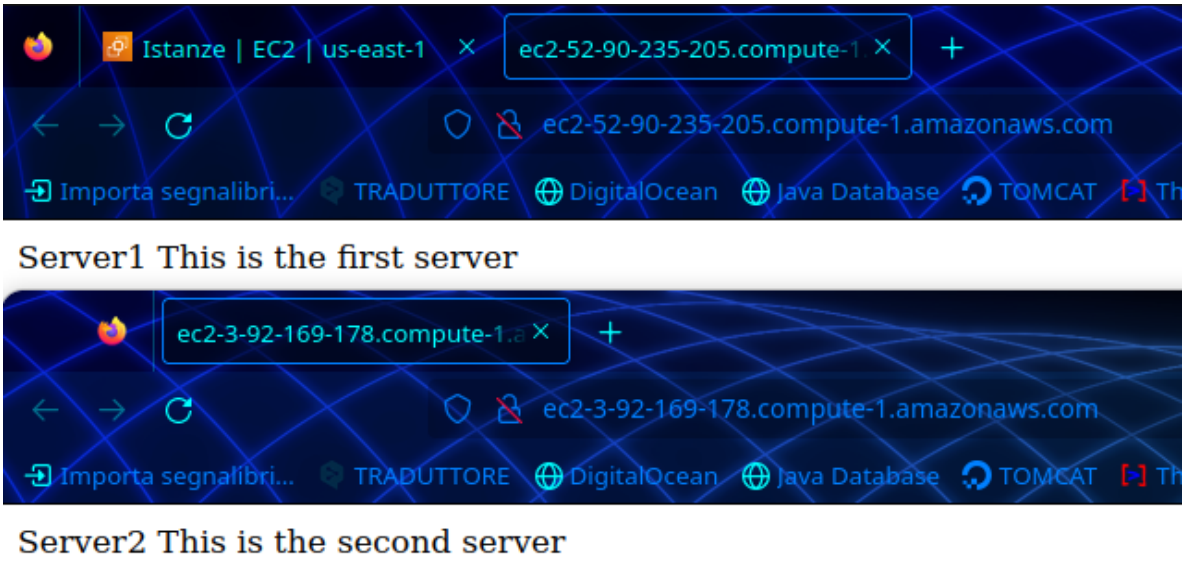
- Deploy two instances on EC2 with a web server that displays a similar page but that can be recognisable as a different server. These servers must be accessible with a browser from the outside.
- Deploy a load balancer that distributes the requests equally between the two servers.
- Prepare an instance template for EC2 to spawn web servers. With the template declare an "Auto-Scaling Group" (ASG) that has a minimum of one instance and a maximum of 2. The ASG must be added to the previously deployed load balancer. Check that the ASG maintains at least one live instance and that the load balancer sends incoming requests to it.
- Extra activity (Optional): Investigate and deploy a database of your choice within AWS. Demonstrate that it works and estimate the cost of usage.

2 Two Servers Web

- Start two EC2 instances as in the previous activity click here to see it by setting the entry rules so that the instances can be accessed from your PC. In addition to port 22, which allows connection via ssh, **port 80** is also enabled so that http requests can be sent.

- Connect to the instance via ssh using the command:
  ssh -i "Key.pem" ec2-user@publicip
- Install httpd:
  sudo yum -y install httpd
- Create a HTML page:
  cd /var/www/html
  sudo nano index.html —— And specify the message on the webpage
- Start httpd:
  sudo systemctl start httpd

- You can now connect through your PC's browser. And by implementing the same procedure with the second instance (where you created a different web page) the result is as follows:



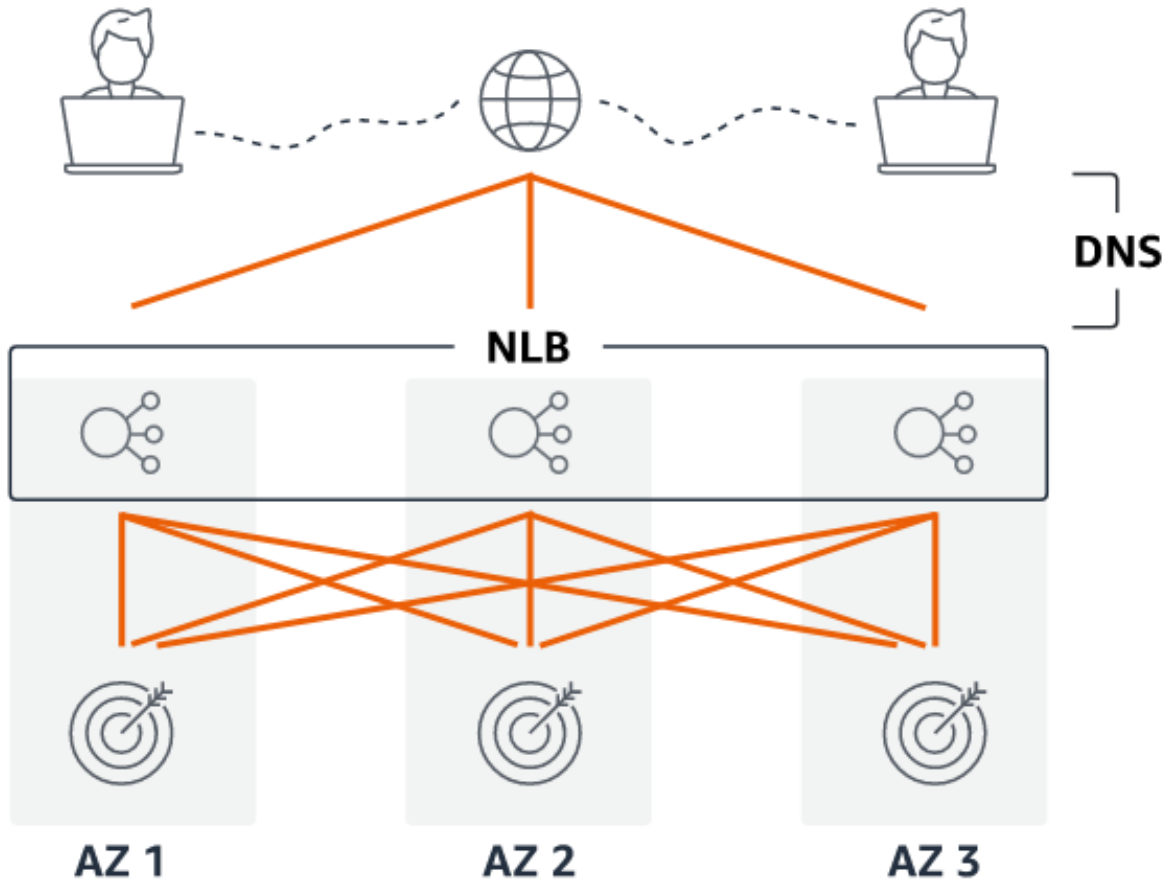
In this way, the first requirement is filled and there are two separate web pages each coming from a different EC2 instance and accessible via SSH and HTTP.

3 Load Balancer

Elastic Load Balancing **automatically distributes your incoming traffic across multiple targets**, such as EC2 instances, containers, and IP addresses, in one or more Availability Zones. It monitors the health of its registered targets, and routes traffic only to the healthy targets. Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

- A **Network Load Balancer** functions at the fourth layer of the Open Systems Interconnection (OSI) model. It can handle millions of requests per second. After the load balancer receives a connection request, it selects a target from the target group for the default rule. It attempts to open a TCP connection to the selected target on the port specified in the listener configuration.
- Choose a name for the load balancer (public) and at least two viability zones.
- Creates a target group with information about the instances.
- Choose a name, instances and procotol.
- Specify two different availability zones.
  (**1d-1c** in the test case).
- Write the path to the web page:
  "**/index.html**"
  This will direct the http request to the previously created index.html page.

- Modify the advanced options to manage this type of control at will. In this case, the client's DNS queries will be resolved to integer **load balancing system IP addresses** in all load balancing system availability zones. Each node of the load balancing system balances the traffic load among the integer targets in all enabled availability zones.
  The image below shows graphically what has just been described:



- After confirming the changes, it will only be necessary to enter the load balancer dns into the browser, which will return the web page as output.
  IMPORTANT
  Httpd must have been started in the ec2 instances.

- By refreshing the page the load balancer will handle the requests:

Server1 This is the first server

As you can see in the address bar, this time there is no address of the EC2 instances but rather of the load balancer.

4 AutoScaling Group

An Auto Scaling group contains a collection of EC2 instances that are treated as a logical grouping for the purposes of **automatic scaling and management**. An Auto Scaling group also lets you use Amazon EC2 Auto Scaling features such as health check replacements and scaling policies. Both maintaining the number of instances in an Auto Scaling group and automatic scaling are the core functionality of the **Amazon EC2 Auto Scaling service**. The size of an Auto Scaling group depends on the number of instances that you set as the desired capacity. You can adjust its size to meet demand, either manually or by using automatic scaling.

An Auto Scaling group starts by launching enough instances to meet its **desired capacity**. It maintains this number of instances by performing periodic health checks on the instances in the group. The Auto Scaling group continues to maintain a fixed number of instances even if an instance becomes **unhealthy**. If an instance becomes unhealthy, the group terminates the unhealthy instance and launches another instance to replace it.

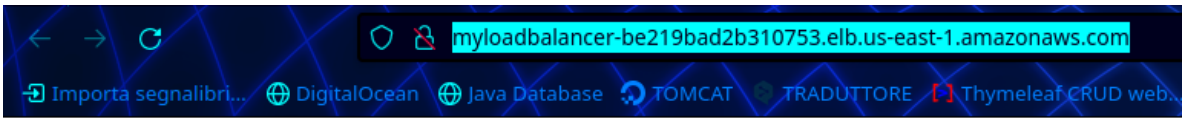
- Open the Amazon EC2 Auto Scaling console and choose:
  Create Auto Scaling group.
- Create an initial template with the desired features
  (In this case we always used **t2.micro** with **AmazonLinux** and the rest of the default settings for the free template).
- Connect to a previously created security group or create a new one.
  ATTENTION
  For the request of this exercise it is necessary that as an entry rule there is TCP with port 80 to allow **HTTP** requests.
- In the user data field enter the script required by the exercise (Create a web page):
  #!/bin/bash
  sudo su
  yum install httpd -y
  systemctl start httpd
  systemctl stop firewalld
  cd /var/www/html
  echo "Test ASG instance" >index.html

  In this way, acquiring administrator privileges, httpd is installed and started. So that the specially made index.html web page can be displayed.
- In the instances section instead we specify a minimum of 1 instance and a maximum of 2 with 2 instances as the preferred value.
- At this point ASG is connected to the load balancer created earlier and after confirming, 1/2 instances will be created which will be visible from the load balancer target group.

Nome	Porta	Zona	Stato di integrità	Dettagli sullo stato di inte...
Server1	80	us-east-1d	Unhealthy	Health checks failed
	80	us-east-1d	Healthy	
	80	us-east-1c	Healthy	
Server2	80	us-east-1c	Unhealthy	Health checks failed

The two red instances are those from the previous exercise that are not currently running, while the two green new ones are the instances created automatically by ASG.

- As in the previous exercise, we will now simply enter the **public address of the load balancer** in the address bar and we will automatically be directed to the web pages of the instances we just created, and the web page created in the launch template will be displayed.



Test ASG instance

Instances created by ASG will be treated by the load balancer in the same way as manually created instances.

5 Costs

- **Load Balancer**
  You are charged for **each hour or partial hour** that a Network Load Balancer is running, and the number of Network Load Balancer Capacity Units (**NLCU**) used by Network Load Balancer per hour. The cost varies by region, in America it is about:
  - 0.0225 dollars per Network Load Balancer-hour (or partial hour)
  - 0.006 dollars per NLCU-hour (or partial hour)

  Details can be viewed at this site.

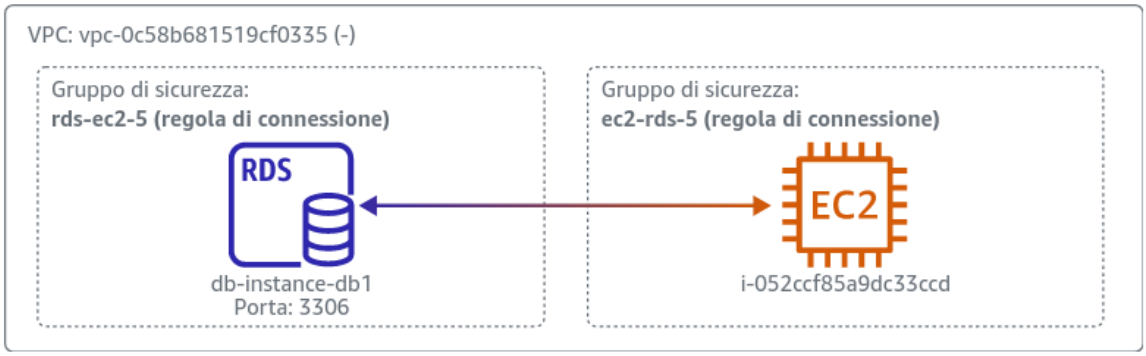
- **AutoScalingGroup**
  There is **no additional charge** for AWS Auto Scaling. You pay only for the AWS resources needed to run your applications and Amazon CloudWatch monitoring fees. So the costs of EC2 instances are the same as those reported in the activity 1 documentation.

6 Extra Activity: RDS

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a **relational database** in the AWS Cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks. To create it:

1. Go to RDS in the AWS Console and click on NEW
2. Select "Standard Creation"
3. Select engine (for example MySQL)
4. Select Free plan
5. Create a **DB Instance** with "name" and "password"
6. Select type of instance and security group
7. Specify the number of port (normally 3306)
8. Select name in additional features
9. Select Create

In order to gain access to the database, a connection can be created with an EC2 instance adding a new **security group** between RDS and EC2:



In this way, a connection can be established from the **EC2 instance** used and the **Database** by following this command (which also shows the related output):

```
[ec2-user@ip-172-31-19-21 ~]$ sudo nc -zv db-instance-db1.cfna3xu1knjn.us-east-1.rds.amazonaws.com 3306
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Connected to 172.31.96.13:3306.
```

If nc is not installed use this command:
sudo yum install -y nc

6.1 RDS Costs

On-Demand database (DB) Instances let you pay for compute capacity **by the hour your DB instance runs**. The cost varies depending on the type of engine (in the exam case, MySQL was used) and it depending on the type of region.
In America it is about (For a single AZ Deployment):

- 0.017 dollars for hour for db.t3.micro instances

Full details are given here.