Getting started with AWS

Objectives

- Gain hands-on experience in using the AWS management console
- Learn how to deploy a simple web application on EC2 and how to configure load balancing and auto scaling for this application

1. Creating a new instance

Sign in to AWS Academy and then go to "Modules" and click on "Learner Lab - Foundational Services". Scroll to the bottom of the Vocareum page and click on "I agree" to accept the agreement. Click on "Start Lab" at the top of the instructions. When the lab is ready, click on the **green** *AWS* button above the terminal window. You should now be connected to the AWS management console. Set the language of the console to English (left side of the bottom navigation bar) to simplify following the instructions.

Select the *EC2* service and click *Launch Instance*. You can now select an Amazon Machine Image (AMI). Choose the *Amazon Linux 2 AMI*. Keep the default option (*t2.micro*) for the instance type.

Select *Create a new key pair*, give a name to your key pair (e.g., *mykeypair*) and click on *Download Key Pair*. Save the key pair to the *.ssh* directory of your home directory (e.g., *~/.ssh/ mykeypair.pem*).

Change the permissions of the key pair:

```
chmod 400 ~/.ssh/mykeypair.pem
```

Click *Launch Instance* and check the status of your instance. When your instance is running (after a couple of minutes), make a note of its *Public IPv4 DNS* address.

You can now connect to your EC2 instance using SSH:

```
ssh -i ~/.ssh/mykeypair.pem ec2-user@{Public_ IPv4_DNS}
```

(for example, ec2-user@ec2-54-227-127-18.compute-1.amazonaws.com)

2. Configuring the instance

Install an Apache HTTP server on the instance and configure it to start at system boot:

```
sudo yum install httpd
sudo service httpd start
sudo chkconfig httpd on
```

Try to access the web server from your local machine by pointing a browser to the public IPv4 DNS address (or the public IPv4 address) of your instance. Is this possible?

Select the security group to which your instance belongs (e.g., *launch-wizard-1*) and add a new inbound rule to allow HTTP traffic from any source. You should now be able to access the web server and see the Apache test page.

Install PHP on the instance and create a simple PHP page:

```
sudo yum install php
cd /var/www/html
sudo nano hello.php
```

Enter the following content and save:

```
<?php echo "Hello! My IP address is: ".$_SERVER['SERVER_ADDR']; ?>
```

Stop and restart Apache:

```
sudo service httpd stop
sudo service httpd start
```

Verify that you can access the PHP page on the instance from your local machine through:

http:// {Public_IP}/hello.php

What is the IP address shown by the PHP page?

In the EC2 Dashboard, stop and then start your instance *(Actions→Instance State→ Stop/Start)*.

Can you still access the PHP page?

3. Load testing the HTTP server

Let's test the performance of your instance using a load testing tool installed in a different machine.

Launch an EC2 instance based on the *Amazon Linux 2 AMI,* connect to the instance using SSH, and install the Apache Bench tool (ab) using:

```
sudo yum install httpd-tools
```

Type the following to run a load test with 10000 requests in total and 5 concurrent requests (simulating concurrent 'users'):

```
ab -n 10000 -c 5 <URL of PHP page>
```

See the output report. Important parameters include: "Requests per second", "Failed requests", "Percentage of the requests served within a certain time".

You can also explore the monitoring metrics of the instance using the AWS console.

Modify the c parameter and find the maximum number of requests per second (with no failed requests) that your HTTP server can support. Identify the maximum c for which 99% of requests are served in under 40 ms.

4. Creating an AMI and another instance

Let's create an AMI based on the HTTP server instance. This AMI will be used as the template for creating additional (clone) instances.

In the EC2 Dashboard, select the HTTP server instance and create an image based on this *(Actions→Image and templates→ Create Image)*. Give a name and description. Once the AMI has been created (look under **Images** in the navigation pane), you can use it to create a new, cloned instance. Click *Launch Instance*, select your AMI from the category *My AMIs*, and use the same key pair. Configure the security group and verify that the PHP page is also accessible on the new instance.

5. Setting up a Load Balancer

We will now set up a load balancer to automatically distribute incoming traffic among our two instances.

In the EC2 Dashboard, under ***Load Balancing***, choose *Target Groups* and click on *Create Target Group*. Give a name for the target group and set the Protocol to *TCP* (instead of *HTTP*). On the *Register Targets* page, select the running instances, click on *Include as pending below*, and create the target group. In the Dashboard, choose *Load Balancers* and click on *Create Load Balancer*. Choose a

Network Load Balancer and give a name. Specify the availability zone and subnet in which the running instances are located. On *Listeners and routing*, select the target group that you have just created, and create the load balancer.

To verify that the instances are ready, choose *Target Groups* in the Dashboard, select your target group and then *Targets*. The state of the instances must become *healthy* (from *initia*l). This step will take a few minutes since the instances must pass three consecutive health checks. Once the instances are healthy, test the load balancer through the URL:

http://{Load_Balancer_DNS_name}/hello.php

Refresh the browser page a few times. What do you observe?

Test the performance of the load-balanced, 2-instance configuration using the EC2 instance in which you have installed Apache Bench. Compare the results with the results obtained previously.

Terminate one of the instances *(Actions→Instance State→ Terminate)* and test the load balancer. What do you observe? Terminate also the other instance.

6. Creating an Auto Scaling Group

We will now use Auto Scaling to manage a set of instances created based on our previously-created AMI.

In the EC2 Dashboard, under **Instances**, choose *Launch Templates* and click on *Create launch template.* Select your previously-created AMI, an instance type (e.g., *t2.micro*) and a security group.

Now, under **Auto Scaling**, choose *Auto Scaling groups* and click on *Create Auto Scaling group.* Select the launch template that you have just created and one of the subnets enabled for the load balancer. On the *Configure advanced options* page, click on *Attach to an existing load balancer* and choose the target group associated with your load balancer. Reduce the *Health check grace period* to a few seconds.

Once the Auto Scaling Group (of capacity 1) is created, verify that a new instance is automatically created and registered as a target of the load balancer.

7. Adding scaling policies

Select your Auto Scaling Group and modify the maximum capacity of the group to 3 (*Details→Edit→Maximum capacity)*. Then, add a scaling policy *(Automatic scaling→ Create dynamic scaling policy)* with policy type *Step scaling* that adds one capacity unit (i.e., instance) to the group when the average CPU utilization is greater than a given threshold (e.g., 50%) for 1 minute. To do this, you will also need to create a new CloudWatch alarm (*By Auto Scaling Group*) for the *CPUUtilization* metric. Similarly, you can add another policy that removes one instance from the group when the average CPU utilization is less than some lower threshold (e.g., 5%).

To verify that the policies work, you can connect to the created instance with SSH and artificially increase its CPU load, for example, using the stress utility:

```
sudo amazon-linux-extras install epel
sudo yum install -y stress
stress -c 1 &
```

You can monitor the CPU utilization (and other metrics) of an instance from the EC2 Dashboard; enable detailed monitoring by selecting the instance and configuring *Actions→Monitor and troubleshoot→Manage detailed monitoring*.

Verify that new instances are automatically created/deleted when the CPU load increases/decreases according to your scaling policies. Note that you may need to refresh the console to see all running instances.

Manually terminate all remaining instances in the auto scaling group. What do you observe?

8. Switching off your resources

Remember that you are being charged for AWS resources. Don't forget to clean up everything (i.e., delete the auto scaling group and the load balancer, terminate any running instances, delete the volumes and snapshots, deregister the AMI, etc.). Click on "End Lab" at the AWS Academy page.

Some links

- *https://aws.amazon.com/getting-started/tutorials/launch-a-virtual-machine/*
- *https://docs.aws.amazon.com/elasticloadbalancing/latest/network/network-load-balancer-getting-started.html*
- *http://docs.aws.amazon.com/autoscaling/latest/userguide/GettingStartedTutorial.html*