Developer Essentials Lab 1

Overview

There are some architectural approaches when using AWS. In this first hands-on laboratory it will be demonstrated how to migrate a monolithic web application that was previously created and deployed on premises.

This approach is called Lift and Shift which consists in:

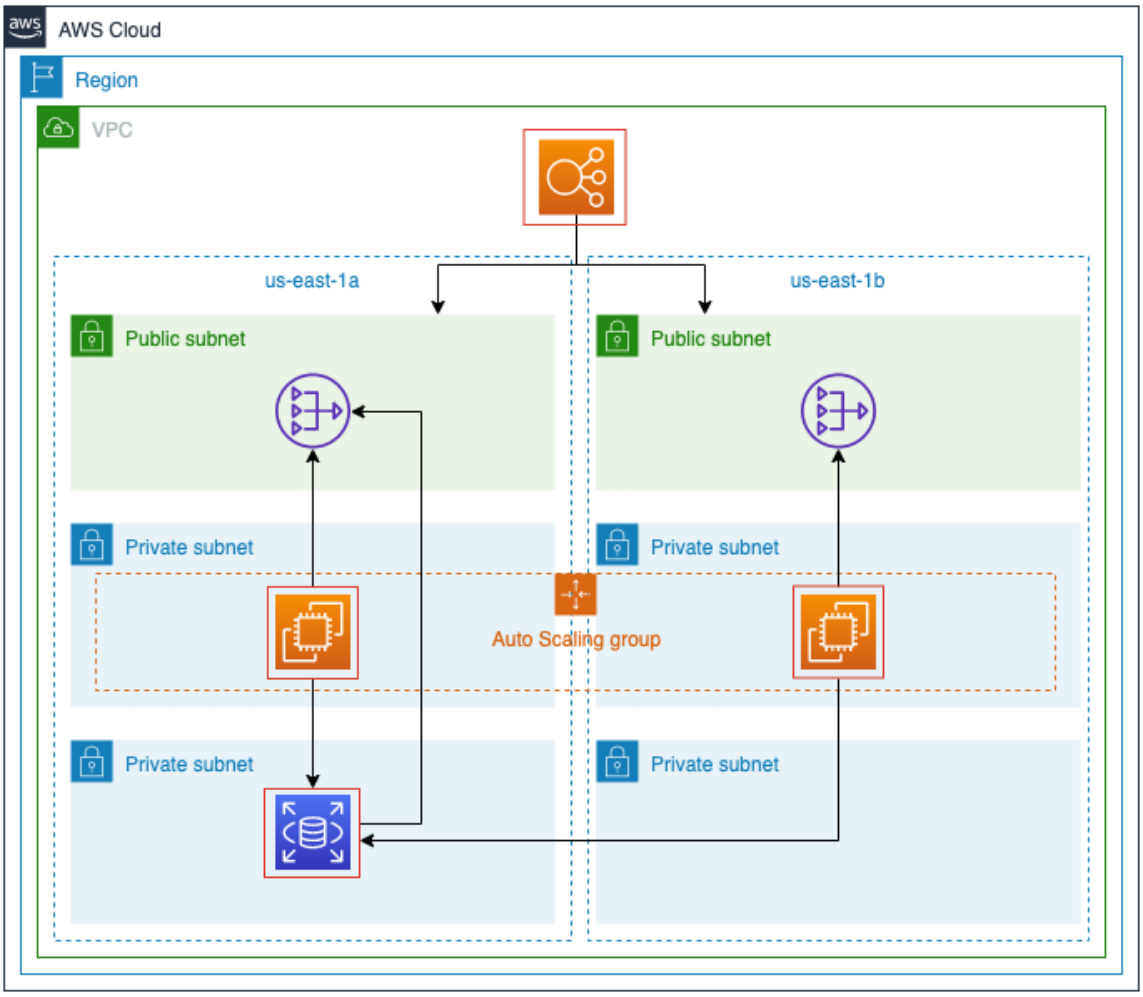
- Deploying existing apps on AWS with minimal re-design;
- Good strategy if starting out on AWS, or if application cannot be re-architected due to cost or resource constraints;
- Primarily use core services such as EC2, EBS, VPC.

Overall Scenario

The laboratory aims to deploy the same application using different architectural patterns.

The code provided represents a monolithic application developed using Java and it performs reads and writes into a relational database, in this case a MySQL engine.

The visual representation of the deployed architecture that you will create on this lab is shown:



You will find several resources already created in the AWS account that you will use during the tasks to make it possible to focus on the application deployment.

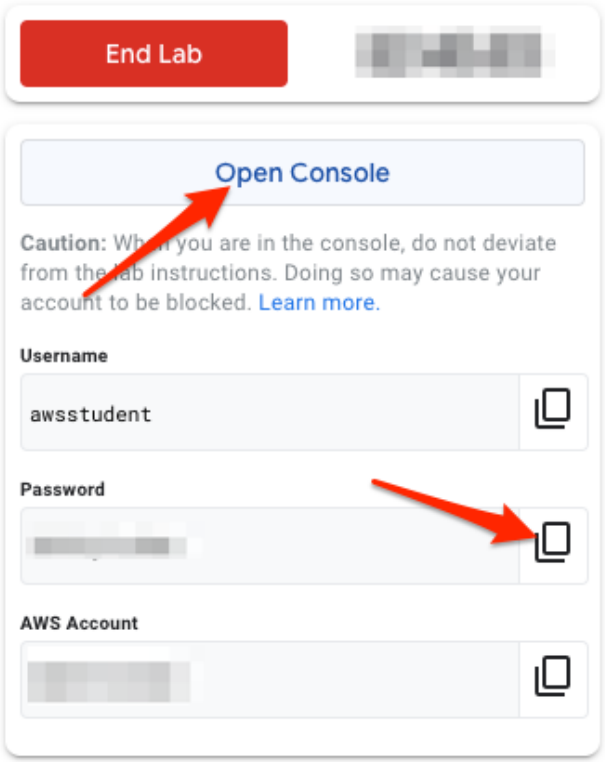
These resources include:

- A custom Amazon VPC with all required components
- A RDS Database Instance using MySQL engine
- Secrets Manager to hold the database credentials
- Necessary IAM Roles and Policies

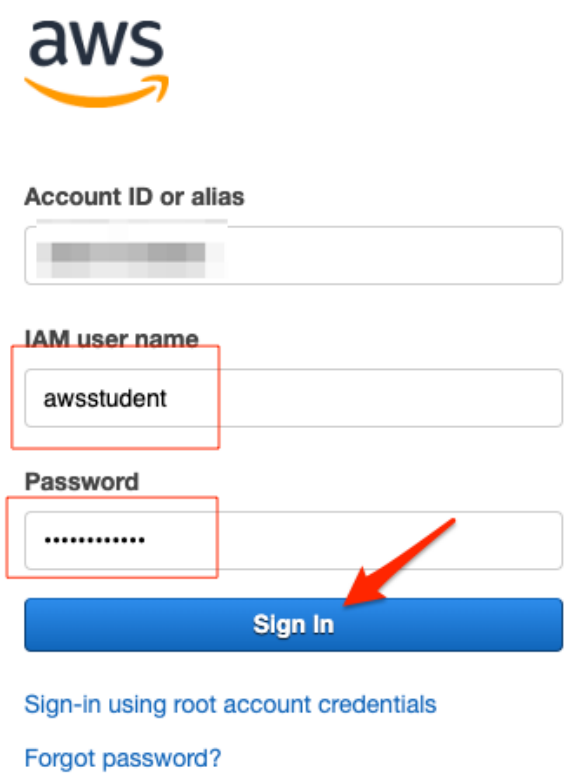
Opening the Lab

1. Click on the **Start Lab** button

2. Wait until the message Provisioning lab resources is gone
3. Copy the password by clicking the button on the right of it
4. Click **Open Console**

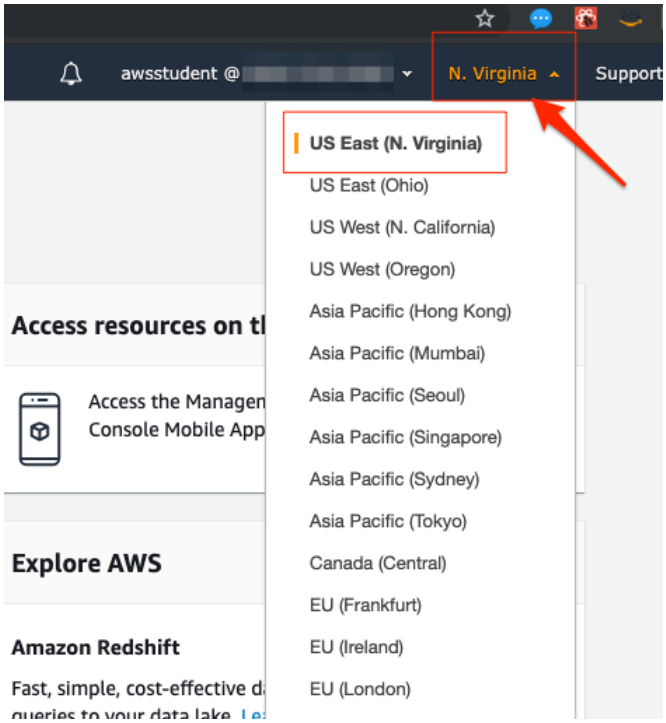


5. Type **awsstudent** as the user
6. Paste the copied password
7. Click **Sign In**



8. Make sure you are in the **us-east-1 N. Virginia** region.

9. You can click on the region upper right portion of the AWS console and choose **US East (N. Virginia)**



Task 1: Deploying the application in a single server

Scenario

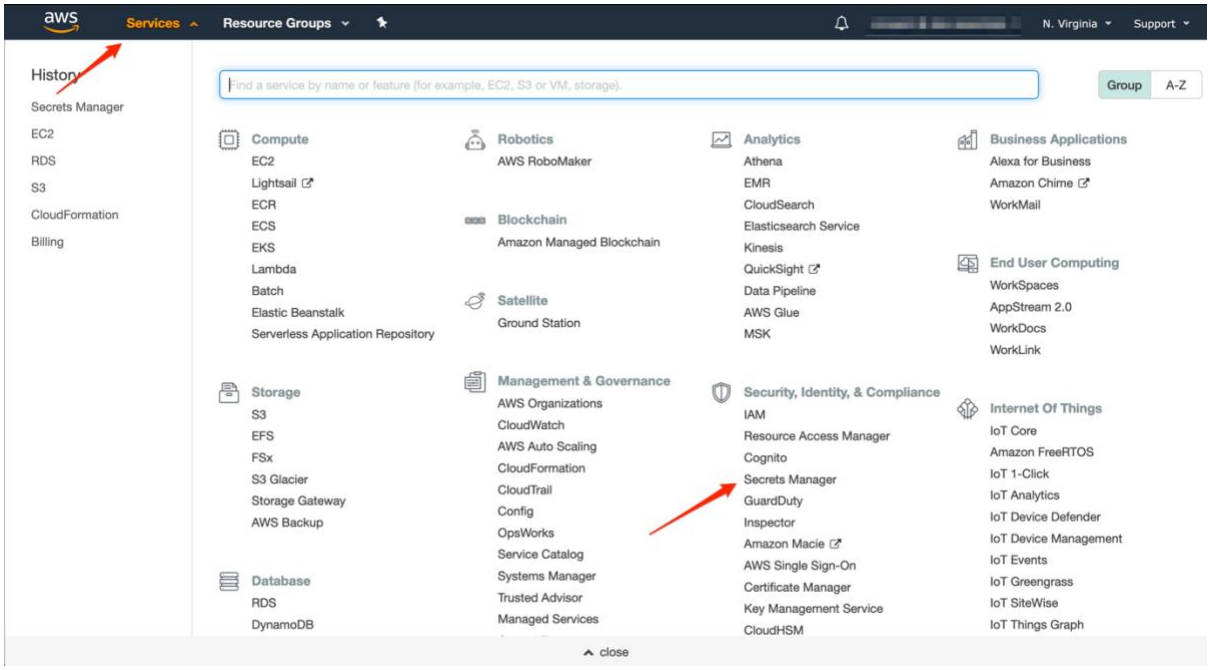
In the context of Lift and Shift scenarios, this is the simplest approach to deploy an application using AWS services.

The goal is to deploy the application into a single Amazon EC2 instance, for that:

- Deploy the application properly by:
  - Getting the database credentials in AWS Secrets Manager;
  - Providing bootstrap commands to the EC2 instance at creation time;
- Test the creation by accessing the application in a web browser

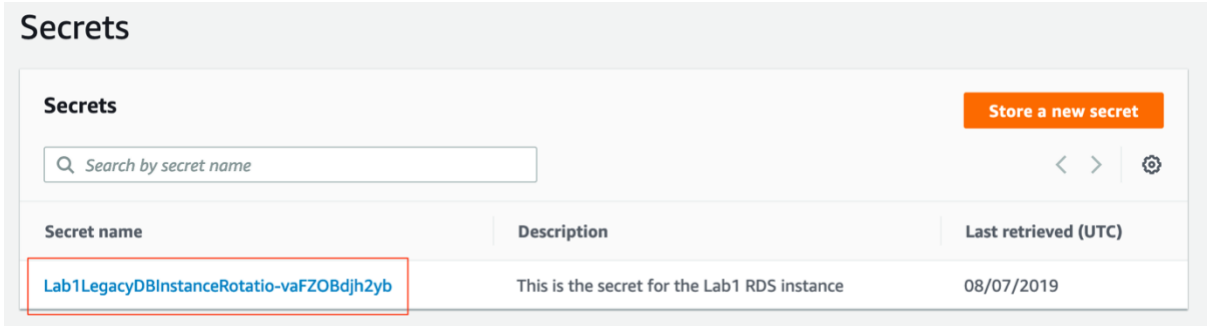
Check the database credentials in AWS Secrets Manager

10. Access the AWS Console, click on **Services** and then choose **Secrets Manager** under **Security, Identity & Compliance**:

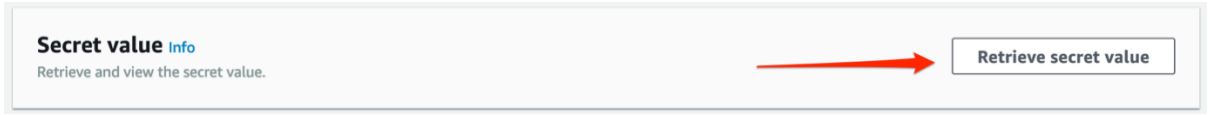


This will open the Secrets Manager console.

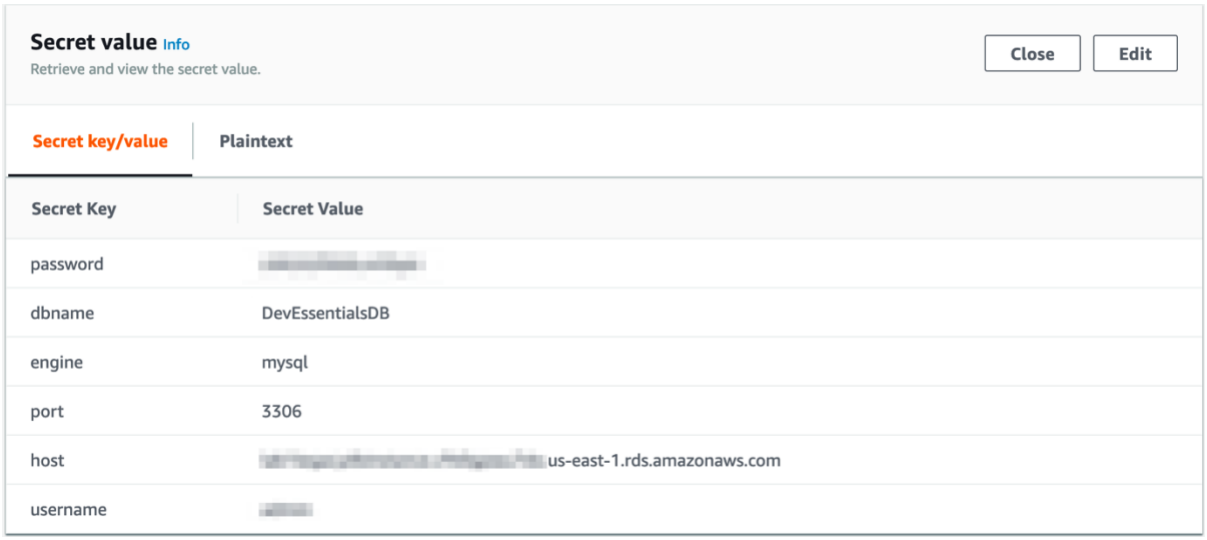
11. Choose the Secret by clicking on the item named **Lab1LegacyDBInstanceRotatio-<HASH>**:



12. Take a moment to review the information in the screen and then scroll down on the page to click on the **Retrieve secret value** inside the **Secret Value** section:



13. The credentials stored for this secret will show.

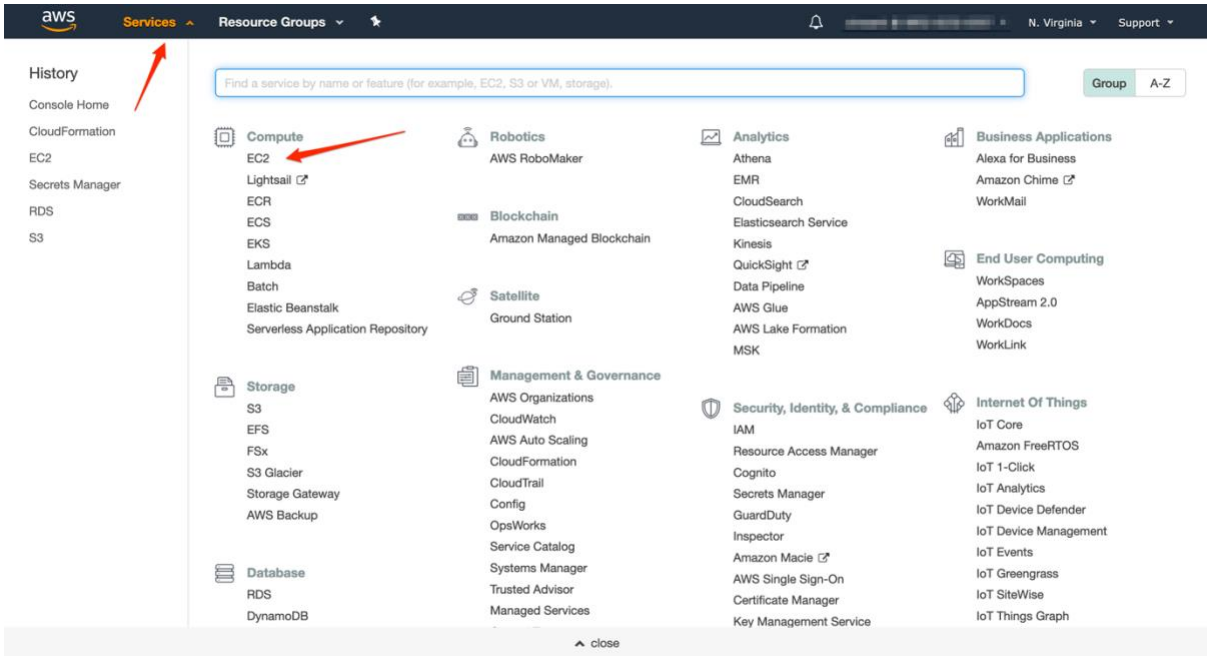


INFO 1: The database credentials set inside Secret Manager where generated automatically on the RDS creation, using a default integration between RDS and Secret Manager. This also enables automatic credentials rotation.

INFO 2: This is for demonstration purposes. In a production environment, this screen would have restricted access for retrieving the secrets information being accessed only via CLI or SDK

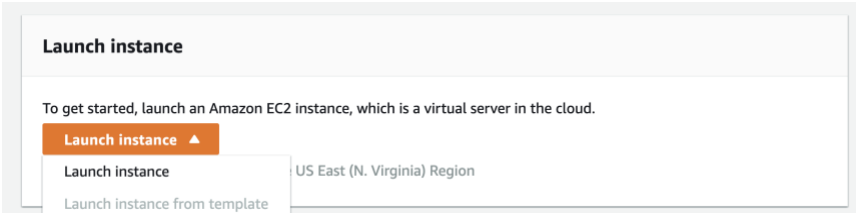
Launch a webserver instance

14. Access the AWS Console, click on **Services** and then choose **EC2** under:

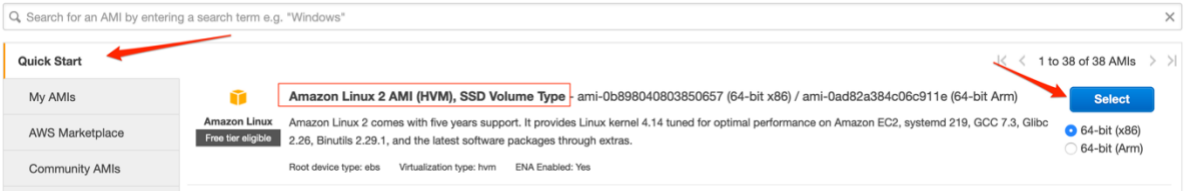


This will open the EC2 console.

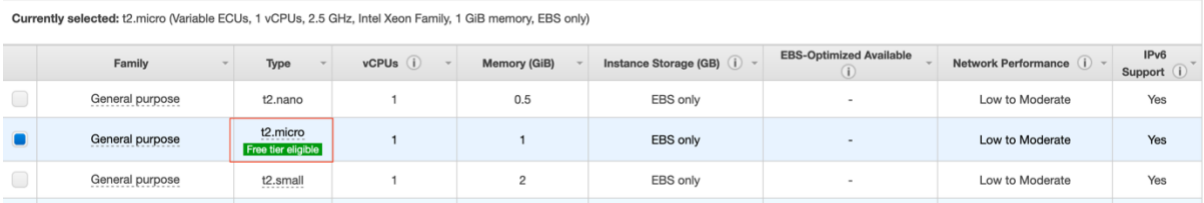
15. Click on **Launch Instance**



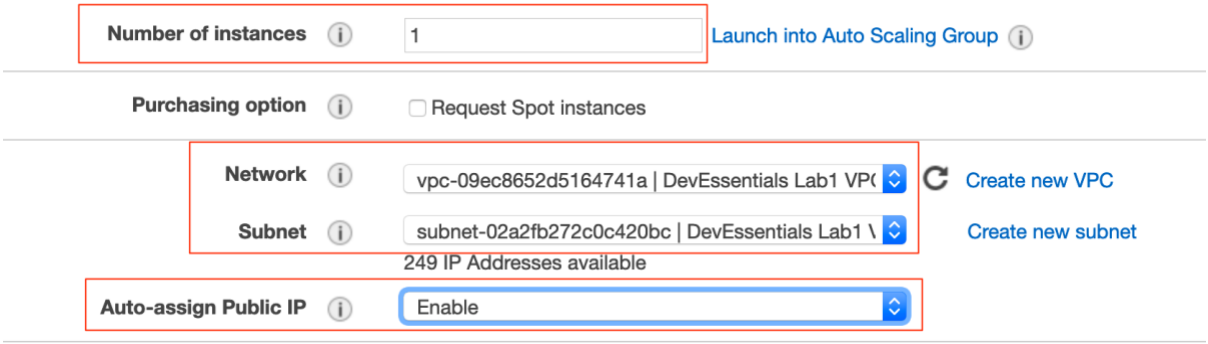
16. In the **Quick Start** section, select the first **Amazon Linux 2 AMI** and click **Select**.



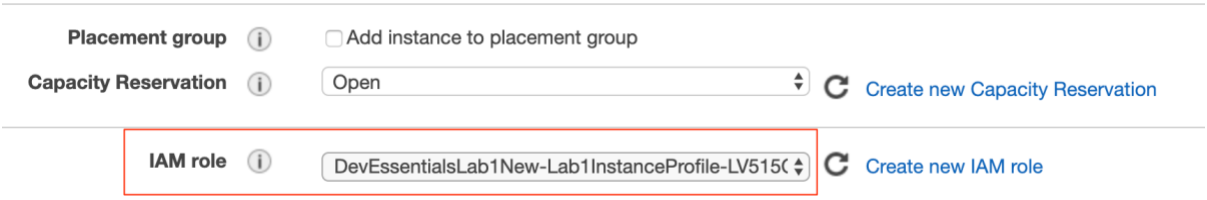
17. In the **Choose instance Type** tab, select the **t2.micro** instance size and click **Next: Configure Instance Details**.



1. Keep the number of instances set to **1**
2. In **Network** settings, select the **DevEssentials Lab1 VPC**
3. Select **DevEssentials Lab1 VPC-public-a** as Subnet
4. Make sure the **Auto-assign public IP** is set to **Enable**



18. In the **IAM Role** field, select **<Stack>-Lab1InstanceProfile-<HASH>**



This will select a role that will grant proper permissions for the EC2 to access other services, like Secrets Manager

19. Scroll down to the **Configure Instance Details** page

20. Expand the **Advanced Details** section

21. Copy/paste the bootstrap commands to the **User data** field:

Get the commands from the official lab 1 instructions in the QuickLabs tool.

The above shellscript does the following actions on the EC2 startup:

- Updates the packages of the instance
- Install the java 8 SDK
- Creates a new directory for the application files
- Download the application package from a S3 bucket
- Get the database credentials programmatically from secrets manager
- Starts the application

Notice the ability to get the database credentials at the runtime, allowing the EC2 instance to query those parameters on-demand in a secured way. The EC2 instance will have the necessary permissions to do so through a IAM role.

22. Click **Next: Add Storage**

23. For the **Storage** step, accept the storage defaults and click **Next: Add Tags**

24. Click the **Add Tag** button and in the **Key** field, set the **Name** value and then in **Value** set to **Single Web Server**. Click **Next: Configure Security Group**

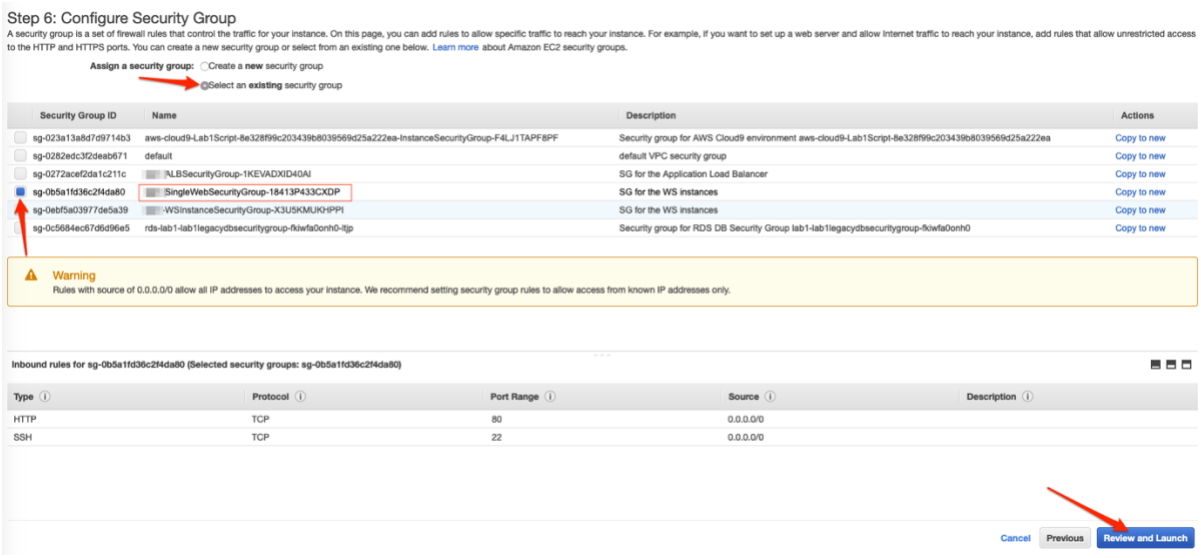
25. In the **Security Groups** section, click on **Select an existing security group**

26. Select the security group with the name **<Stack>-SingleWebSecurityGroup-<HASH>**

27. Don't worry about the warning, we are going to solve this issue in the next part of the lab.

In the real world, your instances should never be opened to the world, specially on the SSH port (22). However in this step, you are executing tests to make you application ready to run in the cloud and you will close this gap in the next steps.

28. Click **Review and Launch**



29. Review your configuration and choices, and then click **Launch**

30. Choose **Proceed without a key pair** and check the "I acknowledge" checkbox.

31. Click the **Launch Instances** button.

32. Click the **View Instances** button in the lower right-hand portion of the screen to view the list of EC2 instances. Once your instance has launched, you will see your Web Server as well as the Availability Zone the instance is in, and the publicly route table DNS name.

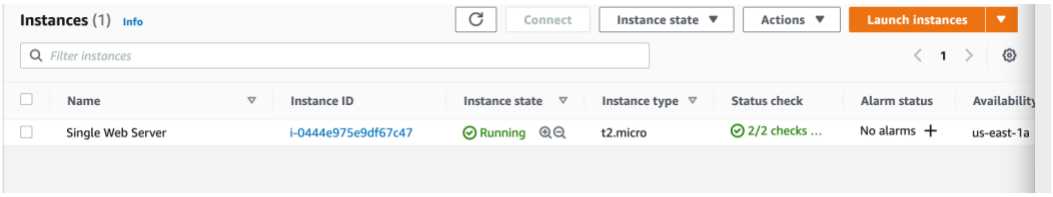
Access the application

33. Make sure that, inside the EC2 console, in the left menu, the **Instances** is selected.

34. Click the checkbox next to your web server (called **Single Web Server**) to view details about this EC2 instance.

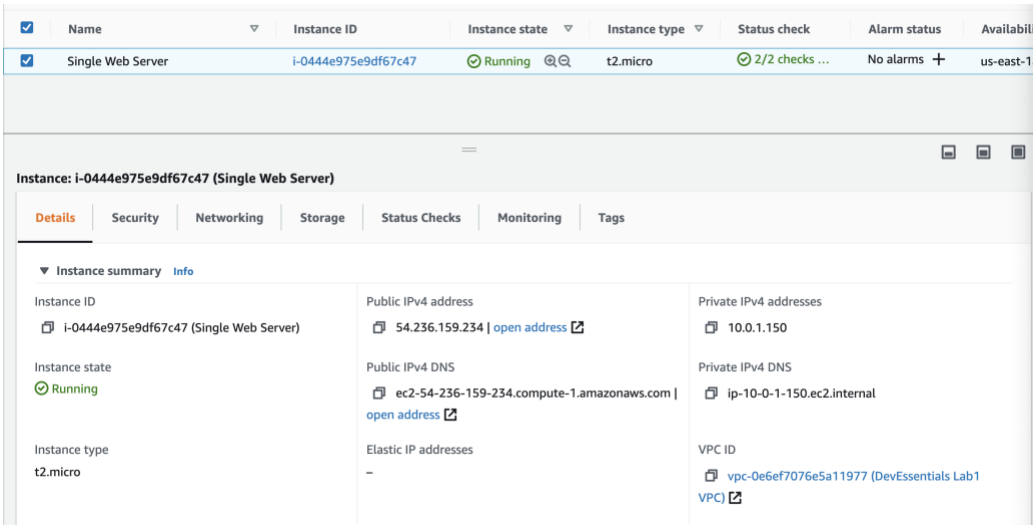
35. Wait for the instance to pass the Status Checks to finish loading.

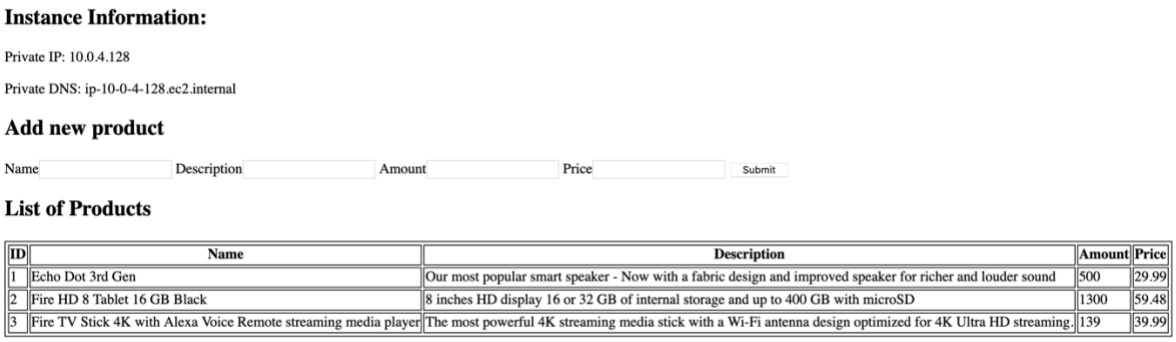
36. Finished initializing



37. In the Instance details on the lower panel, click on the **open address** link in the **Public IPv4 DNS** section.

    Important: when the browser tries to open the application, it is requesting the main page using **HTTPS**. Make sure to change in the URL to **HTTP** only otherwise it won't access the application!





Instance Information:

Private IP: 10.0.4.128

Private DNS: ip-10-0-4-128.ec2.internal

Add new product

Name Description Amount Price Submit

List of Products

ID	Name	Description	Amount	Price
1	Echo Dot 3rd Gen	Our most popular smart speaker - Now with a fabric design and improved speaker for richer and louder sound	500	29.99
2	Fire HD 8 Tablet 16 GB Black	8 inches HD display 16 or 32 GB of internal storage and up to 400 GB with microSD	1300	59.48
3	Fire TV Stick 4K with Alexa Voice Remote streaming media player	The most powerful 4K streaming media stick with a Wi-Fi antenna design optimized for 4K Ultra HD streaming.	139	39.99

You should be able to get some information from the instance and add and list new data into the RDS instance. Try to add a product if you like.

Task 2: Deploying the application with High Availability

Scenario

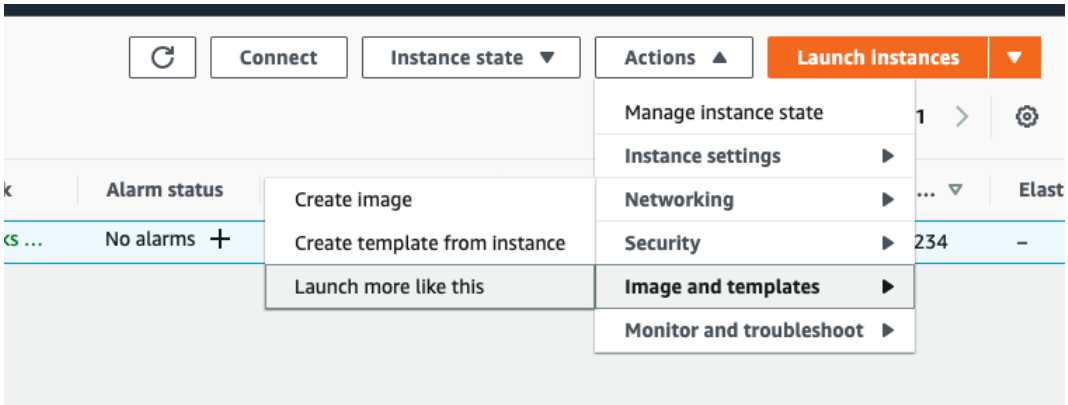
Now there is a deployment up and running of the monolithic application in the cloud. The lift and Shift from on premise is done.

The goal is to have the application deployebd in different availability zones in order to achieve high availability. You will execute the following steps:

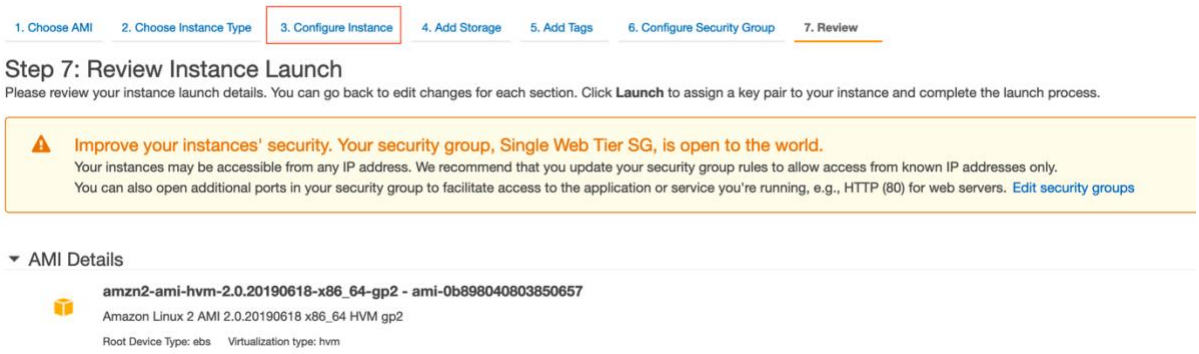
- Create another instance just like the one created in the **Task 1** in a different availability zone.
- Create an Elastic Load Balancer and register both instances as targets
- Access the new architecture using the ELB DNS to serve balanced traffic

Launch a second webserver

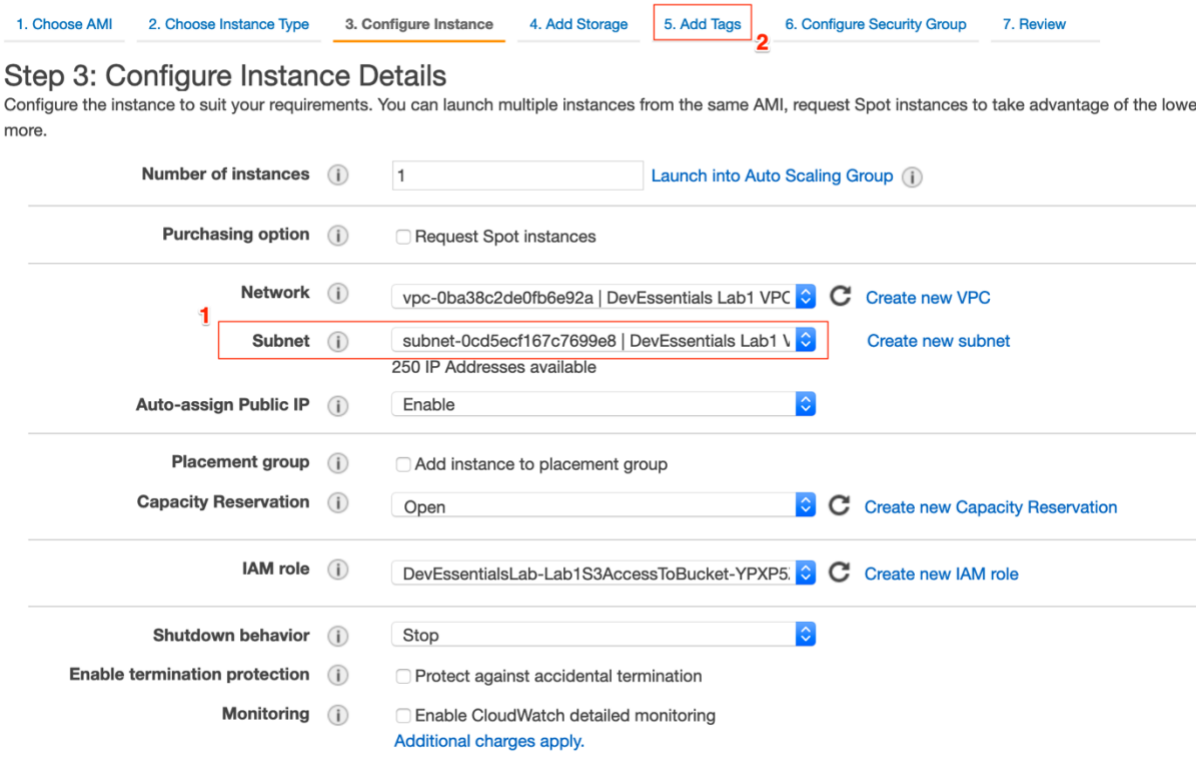
39. In the EC2 console, select the **Single Web Server** instance that is currently running. Click in **Actions** -> **Image and templates** and then **Launch more like this**. This will launch another web server similar to the existing one.



40. Before finalize the instance creation, click in the **3. Configure Instance** tab.

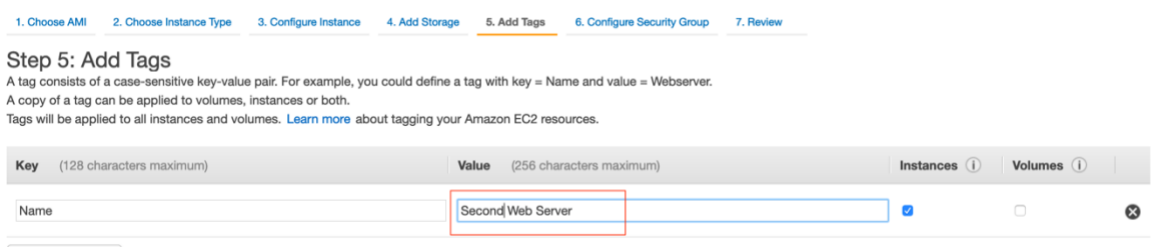


41. In the **Subnet** field, select the **DevEssentials Lab1 VPC-public-b** subnet and then click on **5. Add Tags** tab.



Since the first instance was deployed in the **public-a** subnet, by changing it to the **public-b** subnet will make sure we have instances deployed in different availability zones.

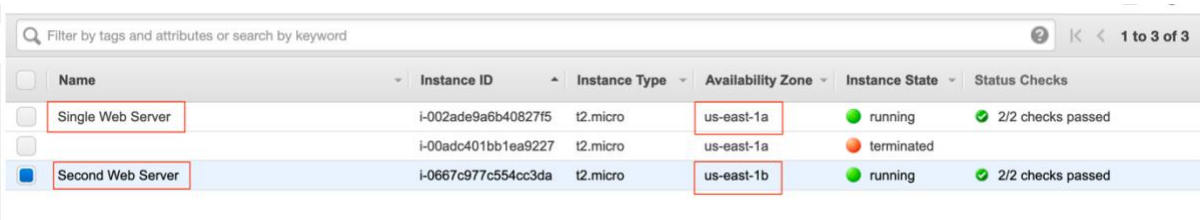
42. Change the tag value to **Second Web Server** and click on **Review and Launch**.



43. In Review screen, review the information and click on **Launch**

44. Choose **Proceed without a key-pair** and then check the "I acknowledge" checkbox. Then click the **Launch Instances** button.

45. Click the **View Instances** button in the lower right-hand portion of the screen to view the list of EC2 instances. You should have 2 instances created by now. Notice the two EC2 instances are provisioned in different Availability Zones.



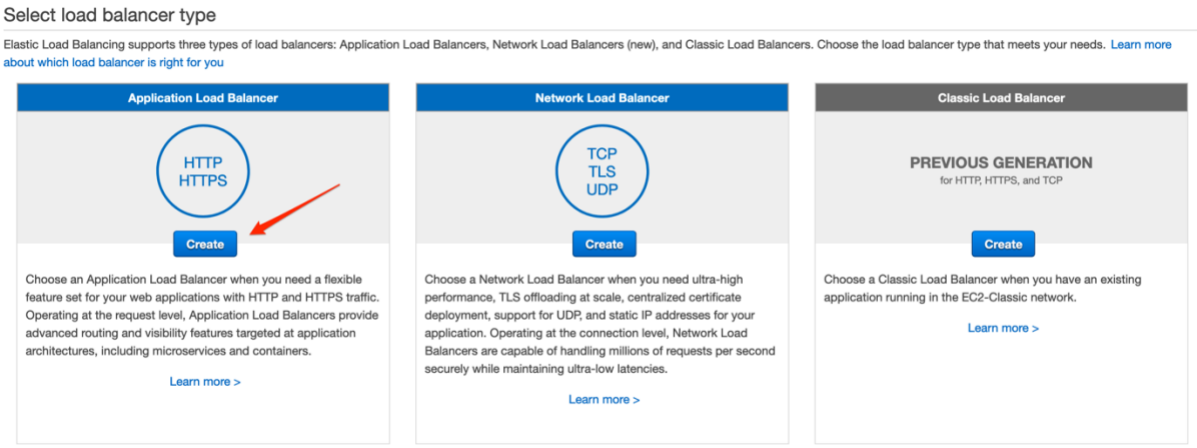
46. Copy the **Public DNS** address for the **Second Web Server** instance and try to open it on a new browser tab. You should see the same webpage application from the first **Single Web Server** instance (don't forget to change to HTTP).

Create an ELB to distribute load

47. In the EC2 console, click on **Load Balancers** in the left menu.

48. Click on **Create Load Balancer** button.

49. Select **Create** for **Application Load Balancer**



50. For the **Name**, enter **Lab1-ALB**. Make sure there is a **Listener** for HTTP on port 80 (It should be already there)

51. As **VPC**, select **DevEssentials Lab1 VPC**

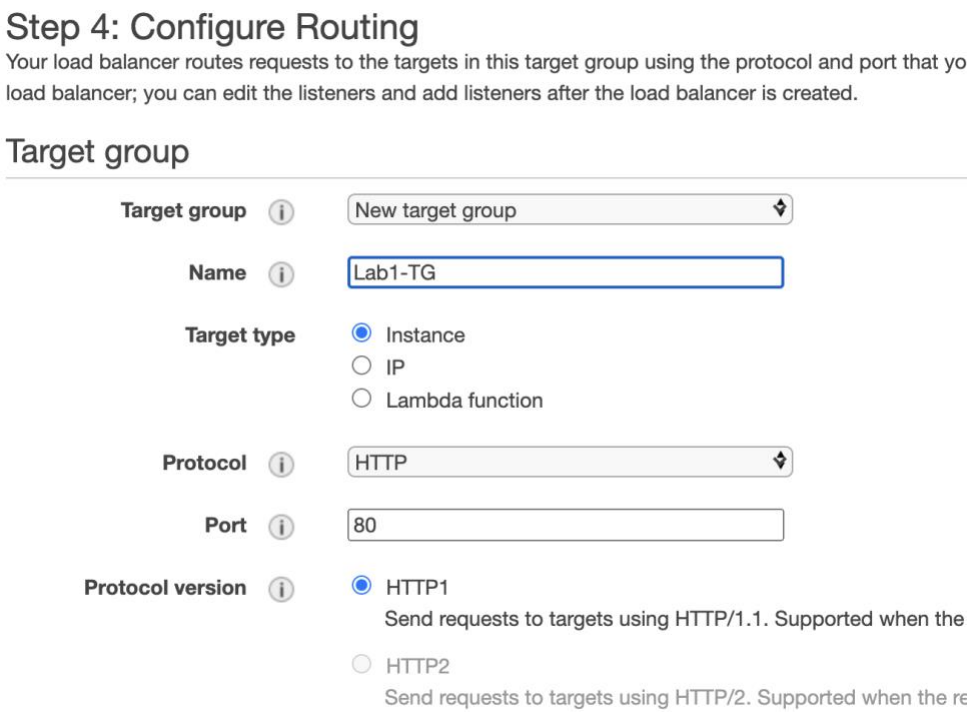
52. In the **Availability Zones** section, make sure to set both AZs which the instances are deployed. Just like the image below. Select the Availability zones and then select subnets **<VPCName>-public-a** and **<VPCName>-public-b**. Finally select **Next: Configure Security Settings**

53. Pass through the next screen by clicking in **Next: Configure Security Settings** and then click on **Next: configure Security Groups**

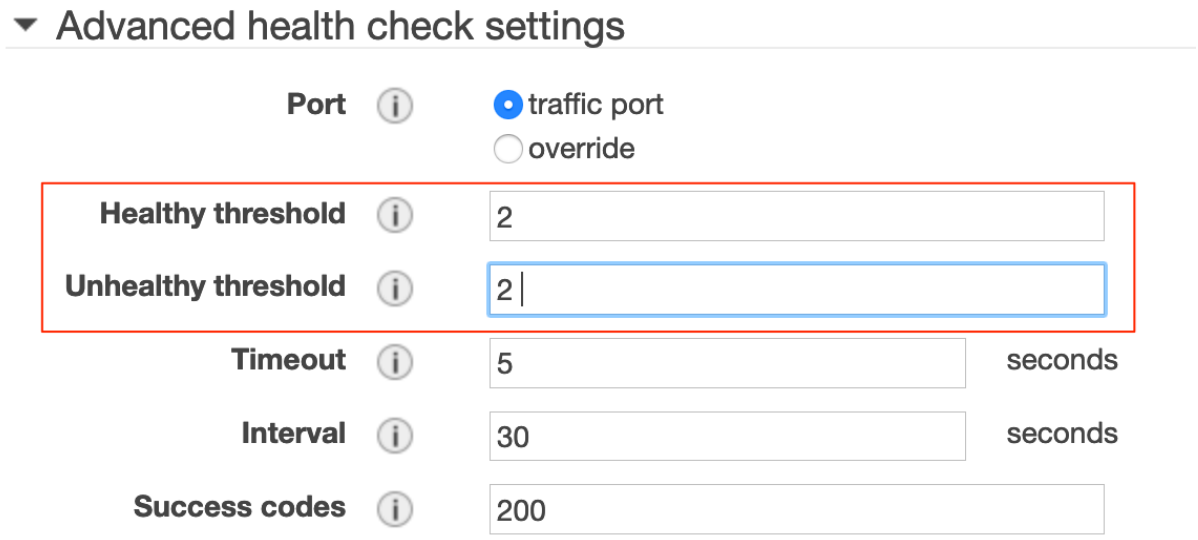
54. Select the existing security group called **<Stack>-ALBSecurityGroup-<HASH>** and click on **Next: Configure Routing**

	Security Group ID	Name	Description	Actions
	sg-010620084aa814989	default	default VPC security group	Copy to new
☑	sg-08b0ac0f5e0983d11	DevEssentialsLab-ALBSecurityGroup-V1375DKNAXEZ	SG for the Application Load Balancer	Copy to new
	sg-0f8280895cf5f08b2	DevEssentialsLab-WSInstanceSecurityGroup-108SHDJ95MIEE	SG for the WS instances	Copy to new
	sg-05effd722aa88d25b	launch-wizard-1	launch-wizard-1 created 2019-08-21T18:06:49.859-03:00	Copy to new
	sg-0f13600d7815f9202	rds-devessentialslab-lab1legacydbsecuritygroup-xacmpjxddgzg-a5b4	Security group for RDS DB Security Group devessentialslab-lab1legacydbsecuritygroup-xacmpjxddgzg	Copy to new
	sg-0a6f3676f3e4419ef	Single Web Tier SG	Single Web Tier SG	Copy to new

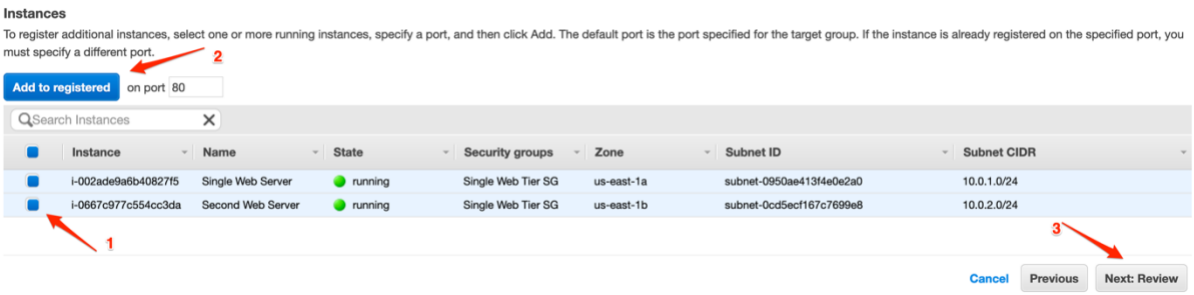
55. For the load balancer to work properly, it is required to create a target group, so give the name **Lab1-TG**



56. Expand the **Advanced Health Check settings** and adjust both **Healthy Threshold** and **Unhealthy Threshold** with **2**. Then click on **Next: Register Targets**



57. Select both your Web Servers you created to add them as targets by clicking on **Add to registered** and click **Next: Review**

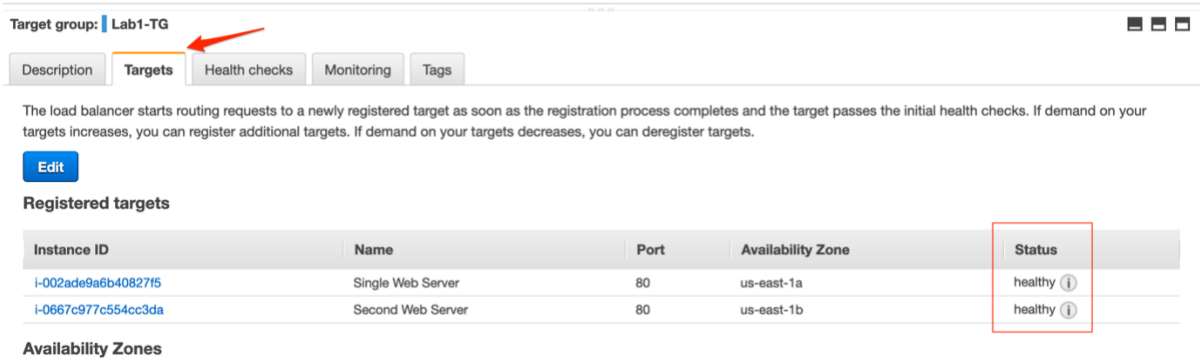


58. Review your configurations and click on **Create**, followed by **Close**.
AWS is now creating your ALB. It will take a couple of minutes to establish your load balancers, attach your web servers, and pass a couple of health checks.

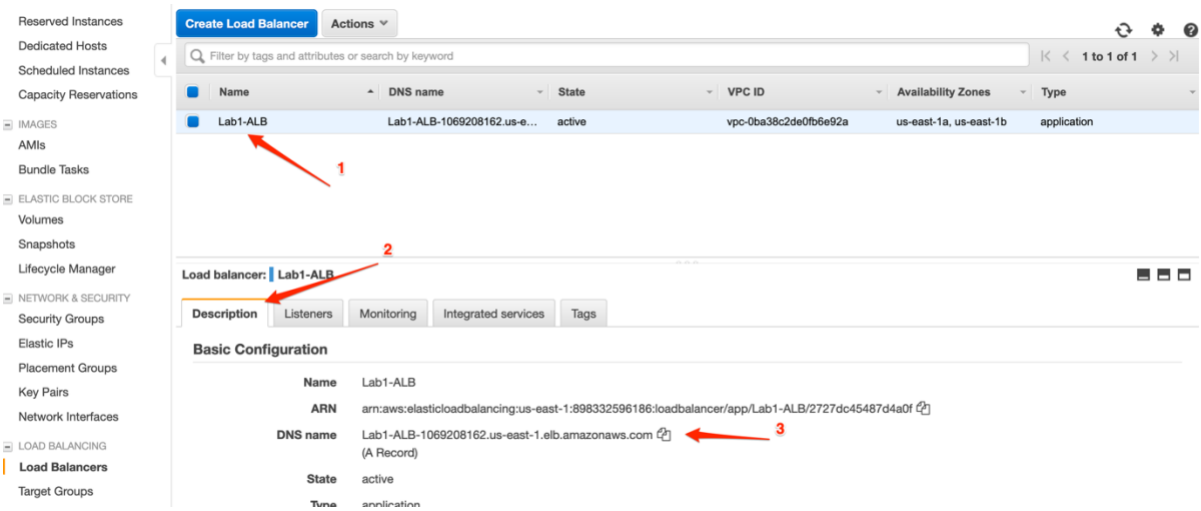
59. To check if everything is working properly, click on **Target Groups** in the left menu.

60. Select the newly created target group **Lab1-TG**

61. On the lower panel for its details, click on the **Targets** tab and monitor the **Status** of the instances until appear **healthy**. This will make sure the Target Group health checks are working properly.



62. In the left menu, click **Load Balancers** link, then select the **Lab1-ALB** and in the tab on the lower panel, copy the DNS value.



63. Open the ALB DNS URL in a new browser tab (incognito mode will help you to prevent browser caching). Hit the browser refresh button and you should cycle through your web servers (you may need to do a “Shift-F5” or “Shift-Refresh” as some browsers like Chrome are pretty aggressive in locally caching web pages). You will be able to see the values for **Private IP** and **Private DNS** fields changing, which means the ALB is serving requests and distributing for the web servers.
    * make sure to use HTTP in the URL you paste in the web browser

First request:

Instance Information:

Private IP: 10.0.2.191

Private DNS: ip-10-0-2-191.ec2.internal

After a couple of refreshes:

Instance Information:

Private IP: 10.0.1.128

Private DNS: ip-10-0-1-128.ec2.internal

Now there is a deployment in 2 different availability zones, which means that you have a highly available architecture, if a problem happens in one of the AZs, the ALB

health check will fail, the connections to the failed instance will be drained and then it will serve requests to the instance in the health AZ.

Task 3: Deploying the application with High Availability, Scalability and Security

Scenario

As the final task to get the most of the lift and shift approach when moving your monolithic application to the cloud, you will apply the scalability concept.

Amazon EC2 Auto Scaling helps you to maintain application availability by scaling your infrastructure as the need or demand arises.
The number of EC2 instances can be scaled in or out as Auto Scaling responds to the metrics you define when creating these groups.

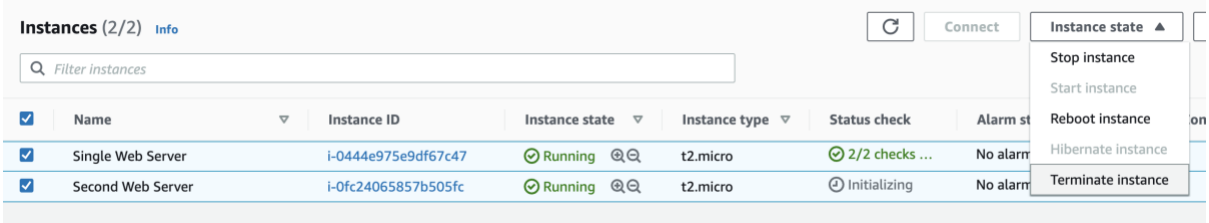
You will execute the following steps in this task:

- Clean up some of the resources created previously
- Create a launch configuration that will be used by the Auto Scaling Group to create instances
- Create an Auto Scaling Group to define the amount of instances to be launched and associate with the Load Balancer
- Test the architecture created

Clean up the created resources

64. Within the EC2 console, in the left navigation pane click on **Instances**.

65. Select the **Single Web Server** and **Second Web Server** instances, click on **Instance state**, then **Terminate instance**

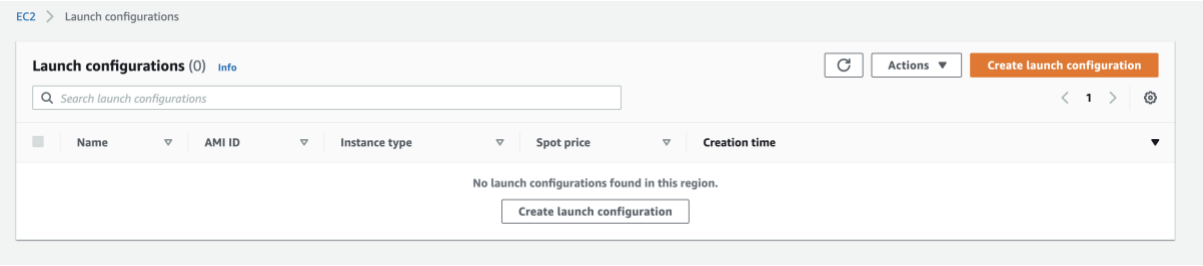


66. Click on **Terminate** to confirm the deletion.

Creating a Launch Configuration

67. Within the EC2 console, in the left navigation pane, find **Auto Scaling** and click on **Launch Configurations**

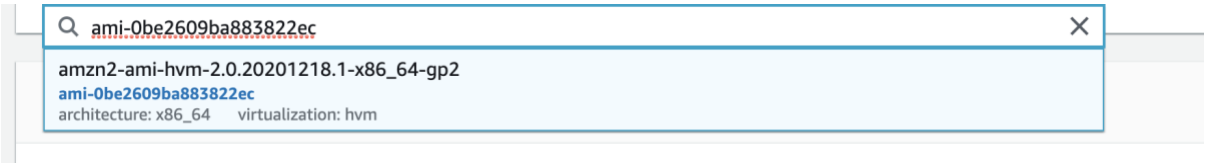
68. Click on **Create launch configuration**



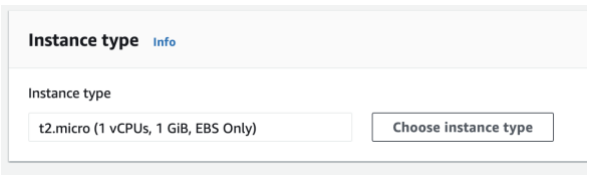
69. For the name, enter **Lab1-LC**

70. In the Amazon machine image (AMI) field, enter exactly the following AMI ID: **ami-0742b4e673072066f**

*make sure no spaces are left before and after the ID when you copy!

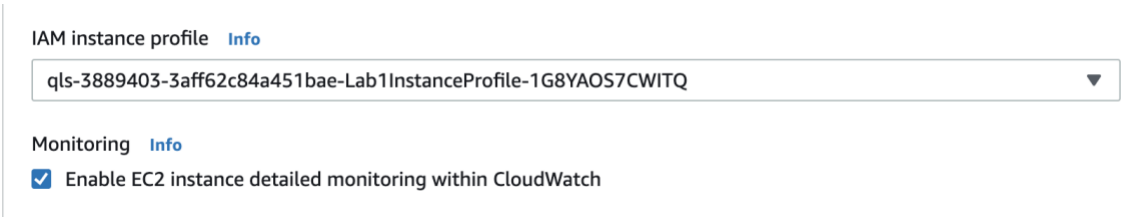


71. For the **Instance type** field type **t2.micro**



72. For IAM **instance profile** select **<HASH>-Lab1InstanceProfile-<HASH>**

73. For **Monitoring,** check **Enable EC2 instance detailed monitoring within CloudWatch**



74. Expand the **Advanced details** and for the **User data** field, copy the same commands used in **step 21.**

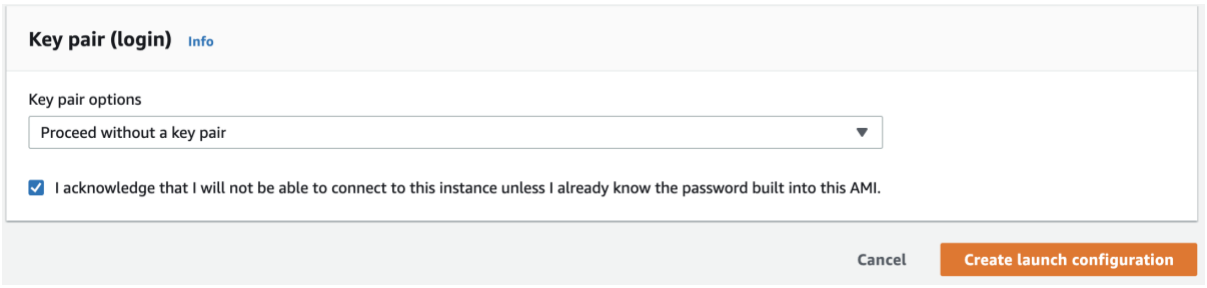
75. For the **IP address type**, make sure to select **Assign a public IP address to every instances**.

76. **Skip** the Storage (volumes) section.

77. Scroll down to **Security groups** section and check **Select an existing security group** then choose **<HASH>-SingleWebSecurityGroup-<HASH>**

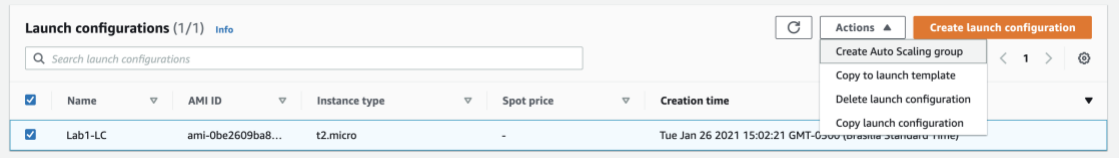
75. Scroll down to the **Key pair (login)** section and choose **Proceed without a key pair** and check **I acknowledge…**

76. Finally click on **Create launch configuration** button.

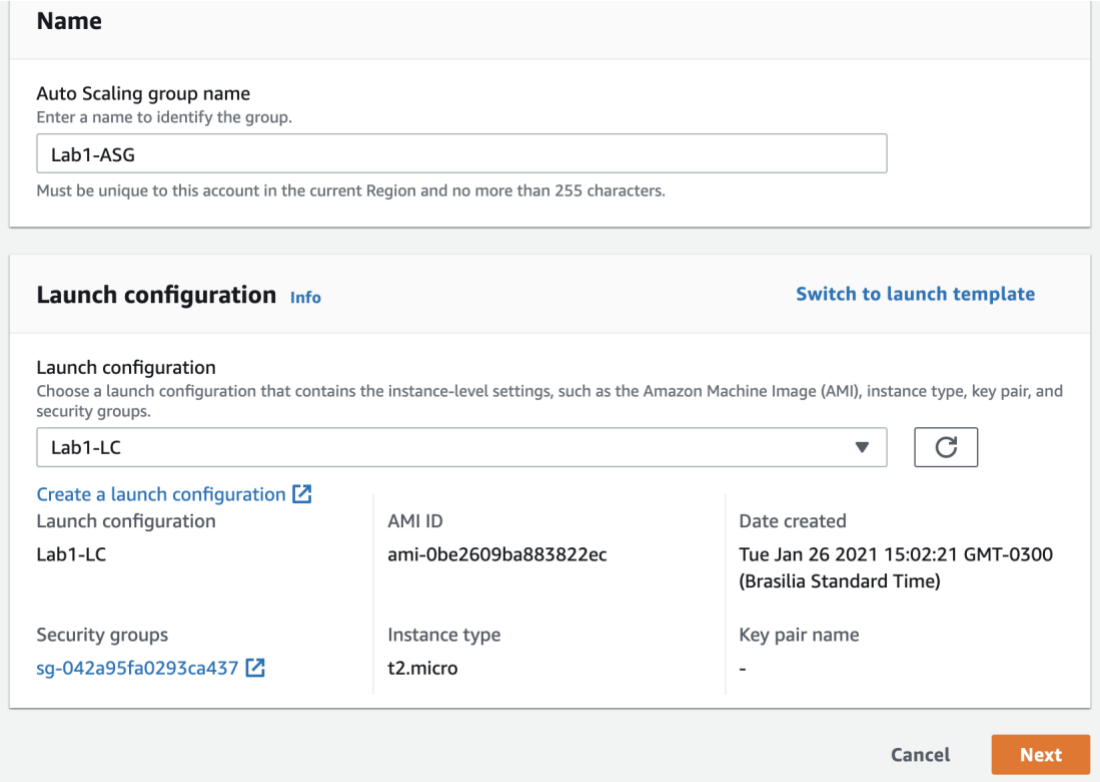


Creating an Auto Scaling Group

77. Within the EC2 console, in the left navigation pane, find **Launch Configurations**, open it and mark the one you just created (Lab1-LC). Select **Actions** and click on **Create Auto Scaling Group**

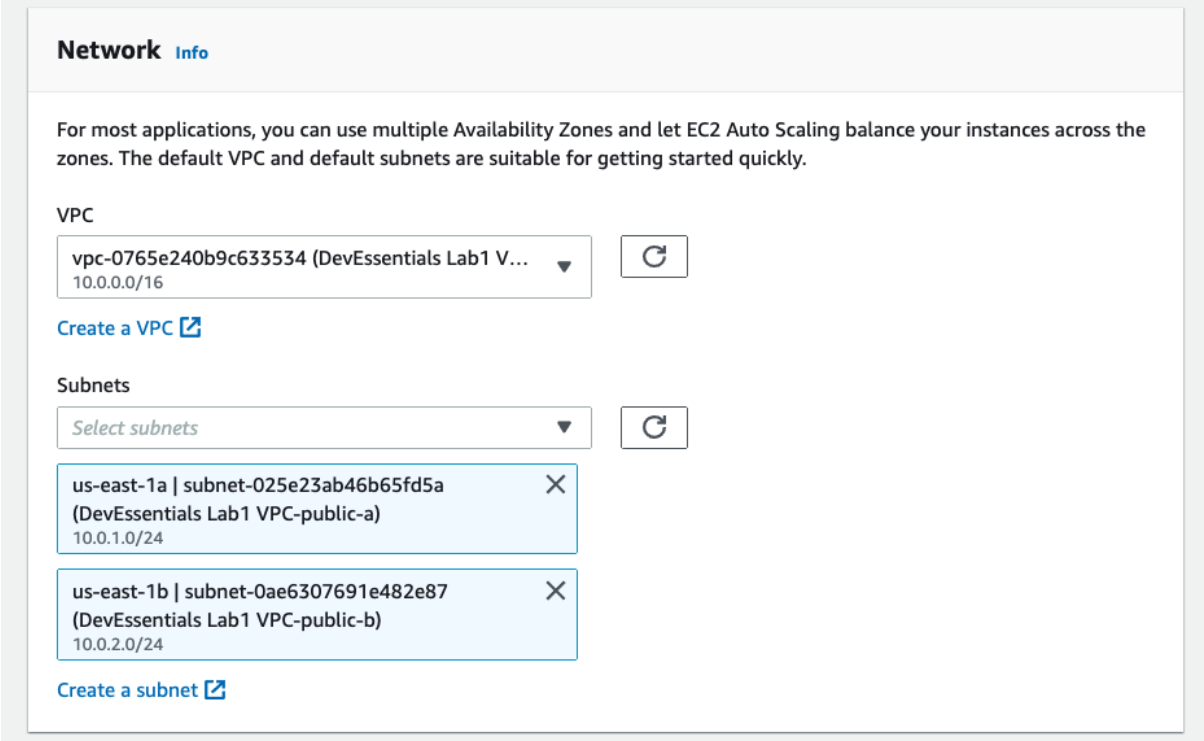


78. For Auto Scaling group name, type **Lab1-ASG.** Click in the **Next** button.

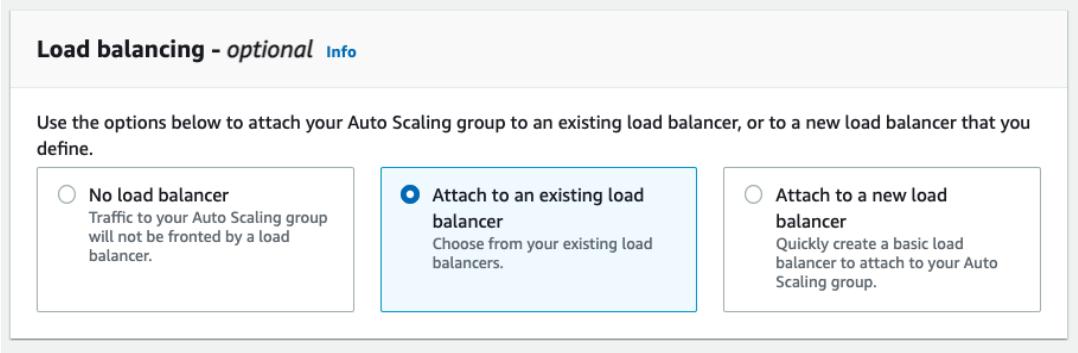


79. Now, in the **Network** section, choose **DevEssencials Lab1 VPC**

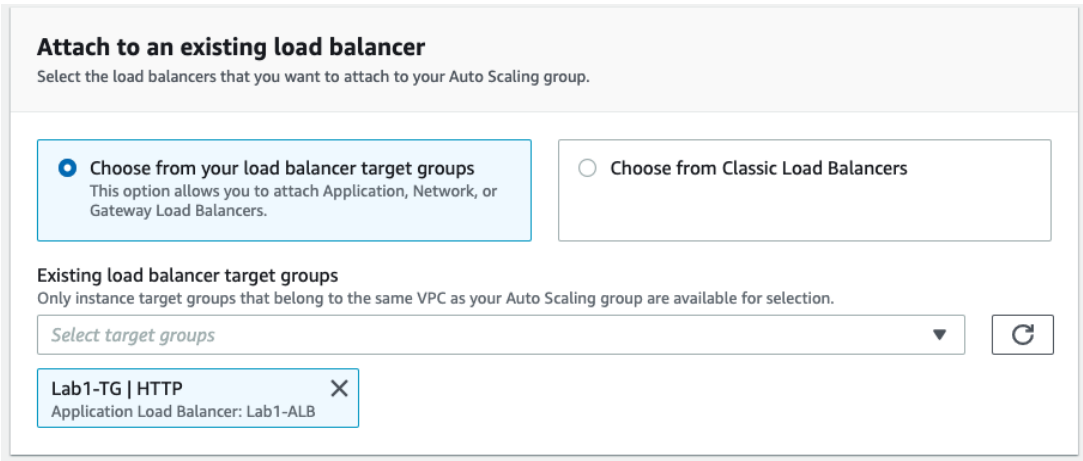
80. For **Subnet**, make sure to select **<VPCName>-public-a** and **<VPCName>-public-b**. Click in the **Next** button!



81. Now let’s attach our Auto Scaling Group to an existing Load Balancer..

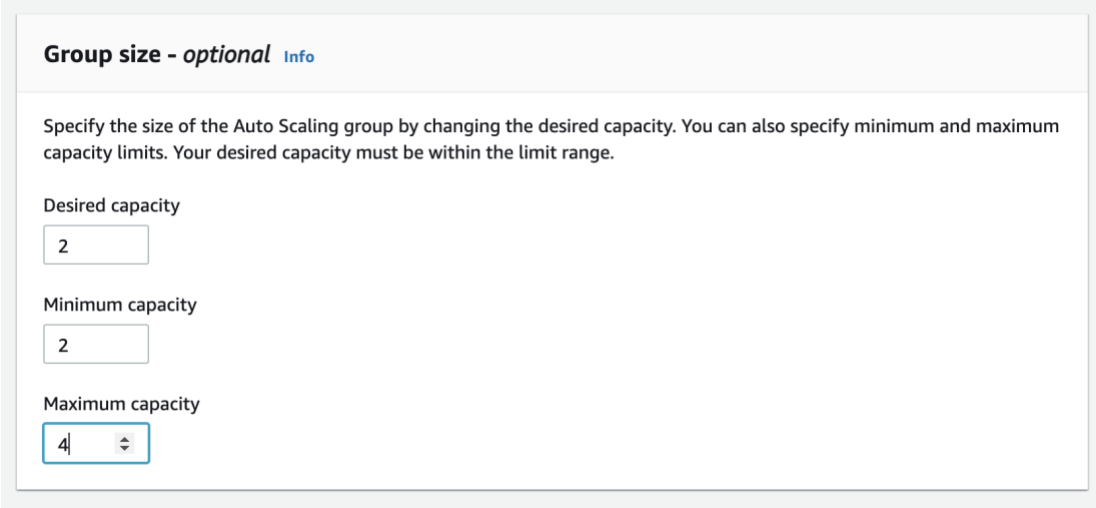


82. Now select your existing target group:

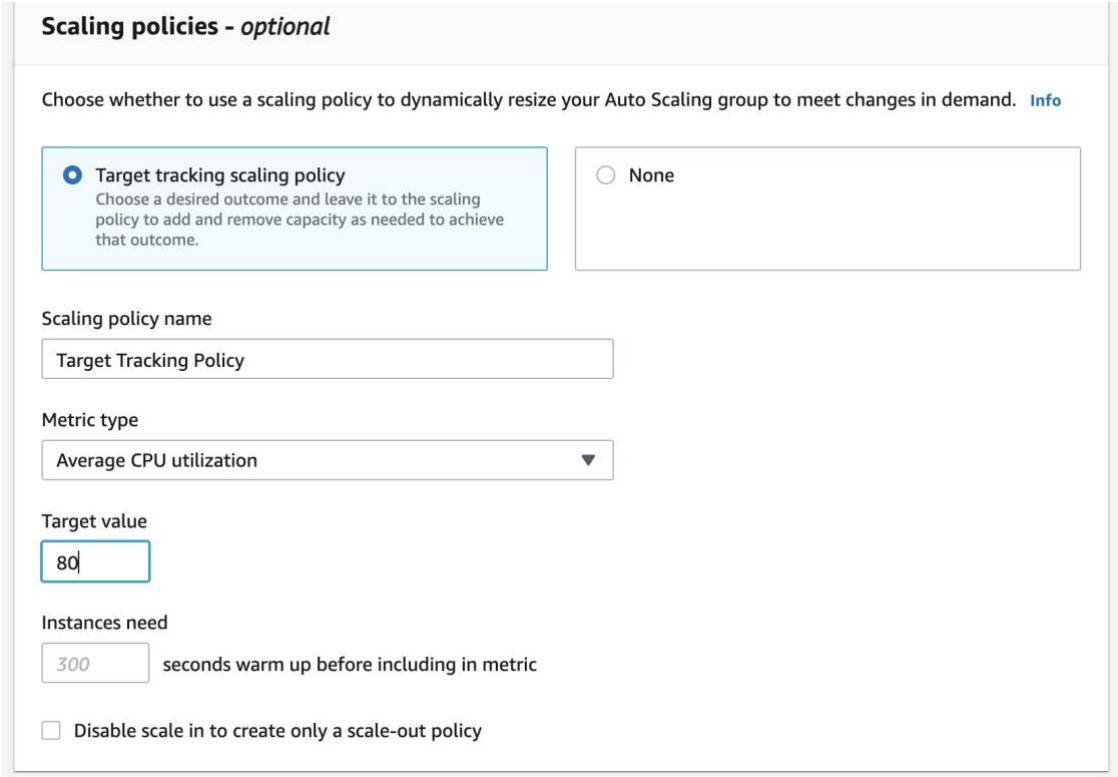


83. Finally click on the **Next** button.

84. Now let's set our group size and scaling policies. Change the Desired capacity to 2, Minimum capacity to 2 and Maximum capacity to 4.

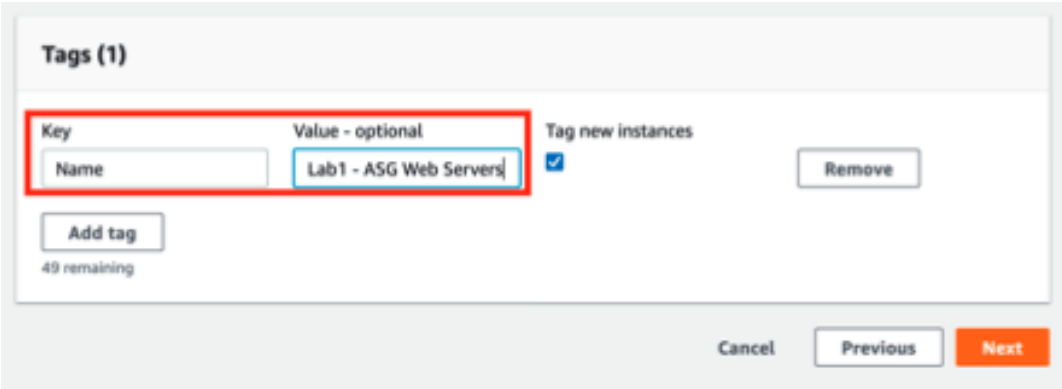


85. Now let's set our scaling policy. It will determine when to increase the size of our Auto Scaling Group. Select the **Target tracking scaling policy** and, for the **Target value**, change to **80**



86. Click on the **Next** button to go to the **Add notifications** section and click again on the **Next** button.

87. Now on the **Add tags** section, let's add the name which will be used as the name of our instances. Add a new tag and type **Lab1-ASG Web Servers**



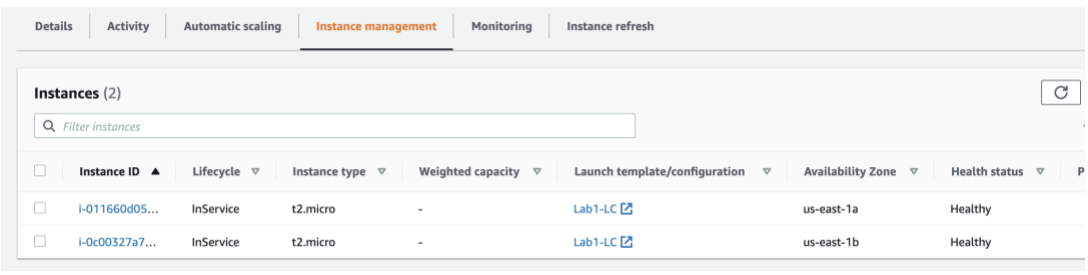
88. Review the whole configuration and click on **Create Auto Scaling group**

Monitor the setup to check if everything was created properly

89. Check the **Auto Scaling Group**. Select the **Lab1-ASG** and pay attention for some considerations:

    - Currently it should be 2 instances running, since it is the desired status.
    - Since the max instances is set to 4, if a running instance reaches 80% of CPU usage, new instances will be created limited to 4 instances.
    - You can make sure High Availability is set, because in the column should be showing **us-east-1a**, **us-east-1b**

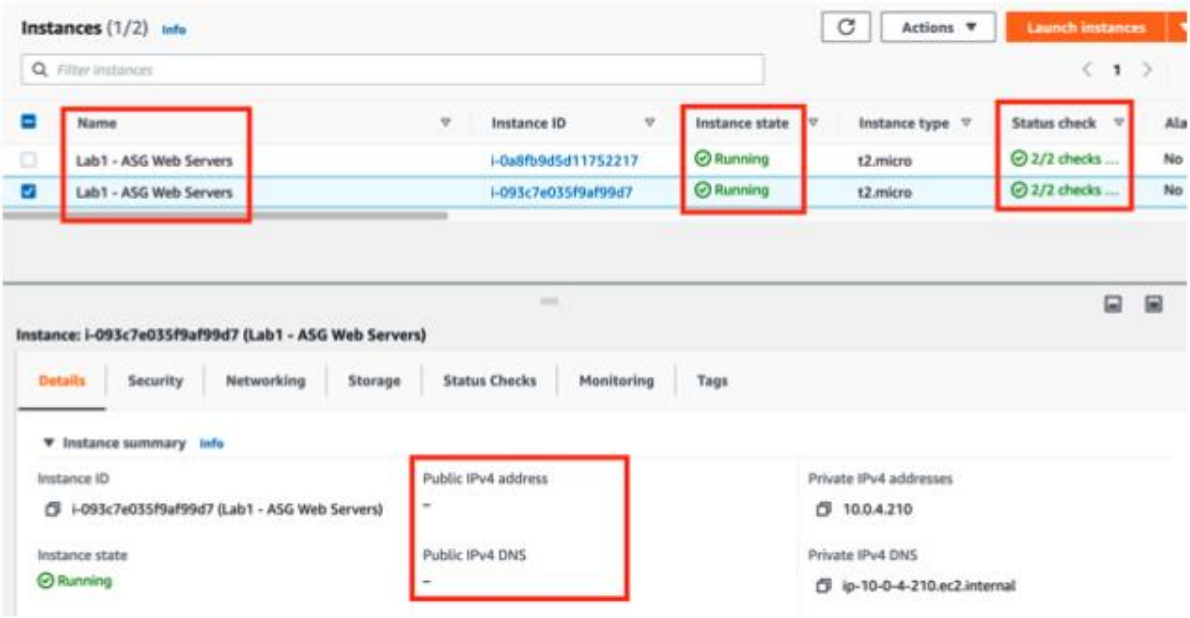
90. Click on **instance management** tab, and check both of the instances are showing **Healthy** in the **Health Status** column. This certifies the instances are running properly.



91. On the left panel, click **Instances** to open the EC2 console.

92. Notice the following details:

- Initially, it should have 2 instances running (the same referenced in the Auto Scaling Group)
- Observe the names are set to **Lab1-ASG Web Servers**, which is the tag set in the Auto Scaling Group setup.
- Make sure that each instance is in a different availability zone
- Make sure the instance state is set to **running**
- Make sure the status checks are showing **2/2 checks passed**
- Make sure there is no Public DNS and no IPv4 Public IP set, there is no direct access to the instances from the internet.



93. On the left panel, under **Load Balancing**, click **Target Groups**

94. Click on **Lab1-New-TG**

95. Click on **Targets** tab on the lower panel and check if the instances were associated properly:

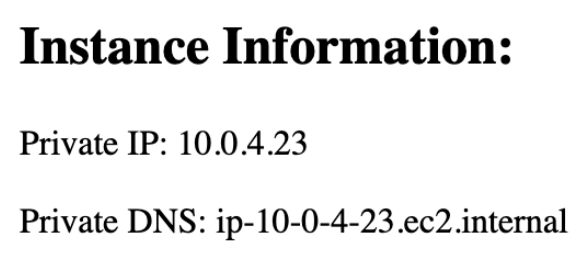
    - Make sure both instances are showing **healthy** in the **Status** column.
    - Make sure the instances are spread in different Availability Zones.

Test the architecture deployed

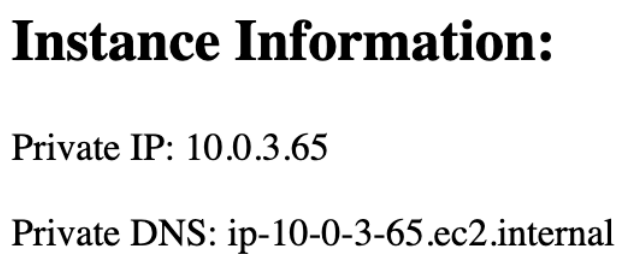
96. Within the EC2 console, in the left navigation pane, find **Load Balancers** and click on the **Lab1-New-ALB**.

97. Copy the DNS name of the ALB and paste it in an incognito tab in your favorite web browser. Hit the refresh button a few times and check the cycle through your web servers.

First request:



After a couple of refreshes:



Congratulations!! You have completed the first hands-on laboratory and you can move on to the next lesson.

Ending Lab

In the , click on the button to release the resources