COS20019

Cloud Computing Architechture

Week 8 – ACA Module 10:
Guided Lab: Creating a Highly Available Environment

Truong Ngoc Gia Hieu
105565520

Guided Lab: Creating a Highly Available Environment

A. Lab Overview and objectives

Critical business systems should be deployed as highly available applications; that is, applications remain operational even when some components fail. To achieve high availability in Amazon Web Services (AWS), you run services across multiple Availability Zones.

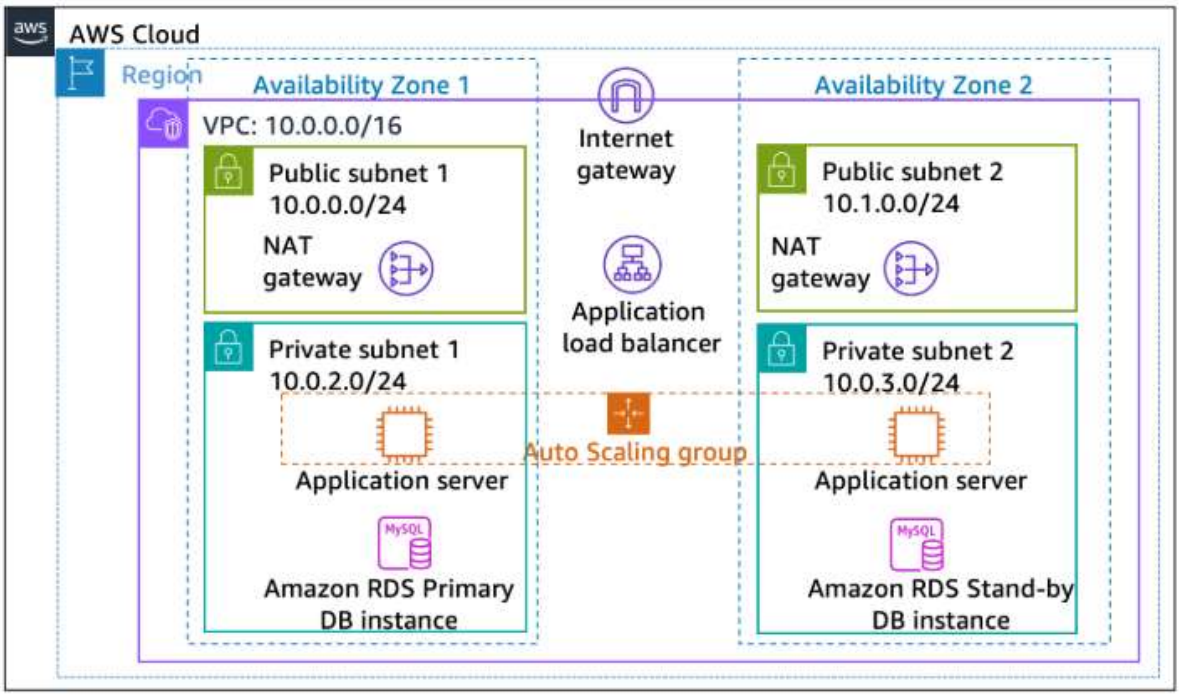
Many AWS services are inherently highly available, such as load balancers. Many AWS services can also be configured for high availability, such as deploying Amazon Elastic Compute Cloud (Amazon EC2) instances in multiple Availability Zones.

In this lab, you start with an application that runs on a single EC2 instance and then make it highly available.

After completing this lab, you should be able to do the following:

- Inspect a provided virtual private cloud (VPC).
- Create an Application Load Balancer.
- Create an Auto Scaling group.
- Test the application for high availability.

At the end of this lab, your architecture will look like the following example:



Duration

The lab requires approximately **40 minutes** to complete.

AWS service restrictions

In this lab environment, access to AWS services and service actions might be restricted to the ones that are needed to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond the ones that are described in this lab.

Accessing the AWS Management Console

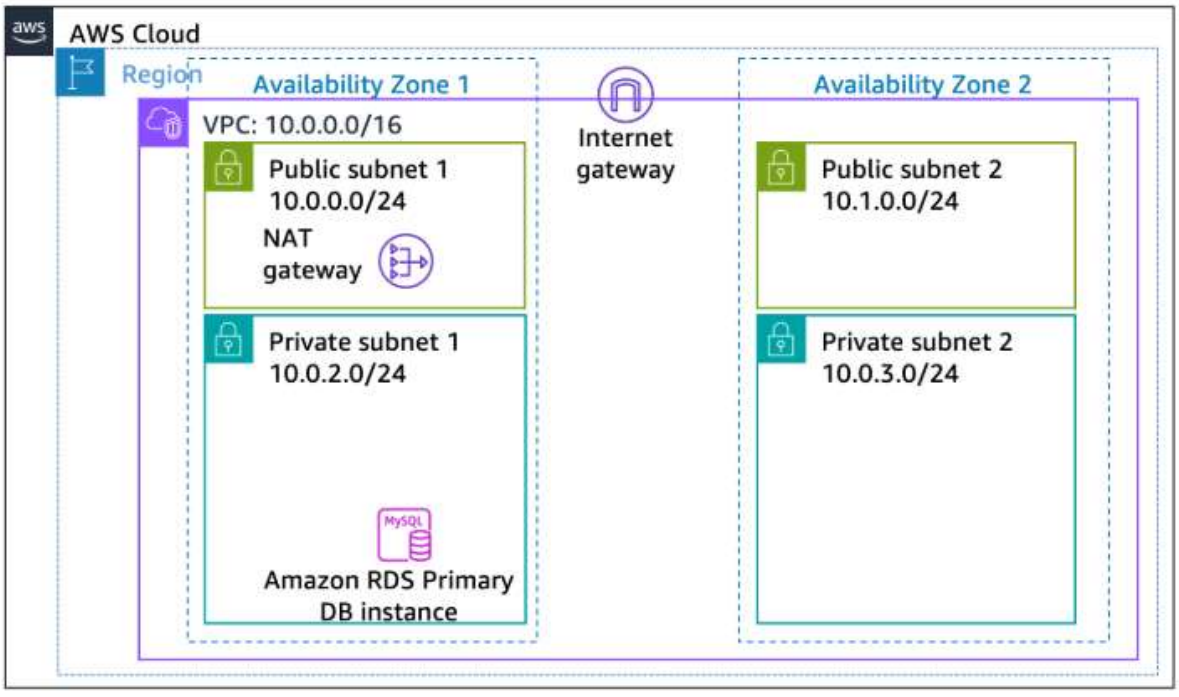
To access the **AWS Management Console**, click on the button **Start Lab** and wait until the **circle symbol** next to the AWS **turns from yellow to green.**

Figure 1: AWS Activated

B. Task 1: Inspecting your VPC

This lab begins with an environment that is already deployed through AWS CloudFormation. This environment includes the following:

- A VPC
- Public and private subnets in two Availability Zones
- An internet gateway that is associated with the public subnets
- A NAT gateway in one of the public subnets
- An Amazon Relational Database Service (Amazon RDS) instance in one of the private subnets



Step 1.1: After accessing the AWS Management Console successfully, search and choose **VPC** to open the Amazon VPC console.

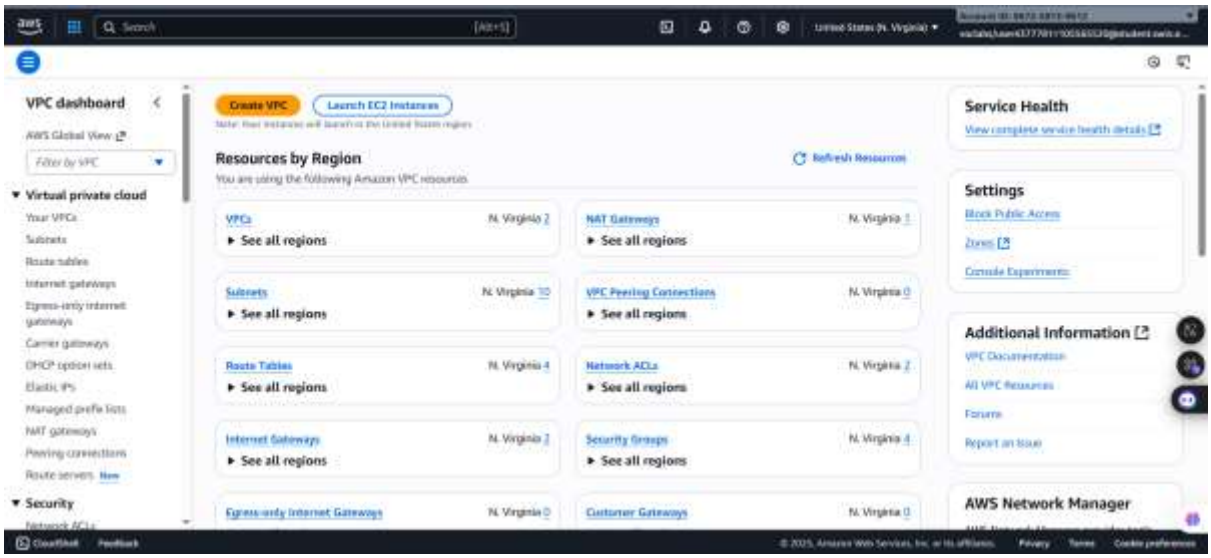


Figure 2: VPC Console page

Step 1.2: In the left navigation pane, under the **AWS Global View**, press the arrow pointing down in the box which has text "**Filter by VPC**" and then select **LabVPC.**

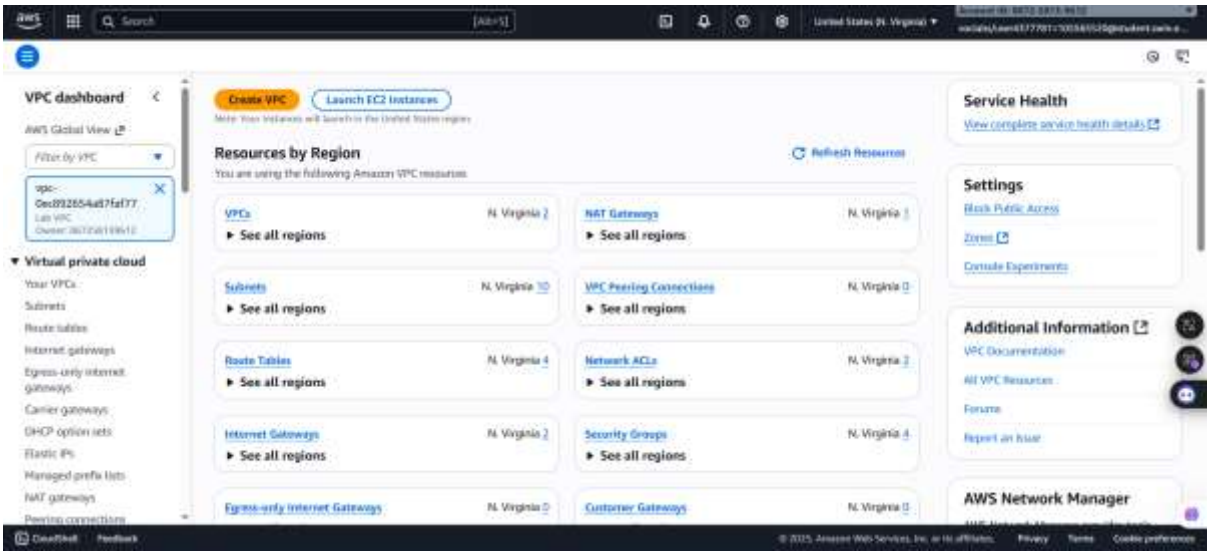


Figure 3: Filter VPC

Step 1.3: Next, choose Your VPCs in the left naviagtion pane also to view VPC

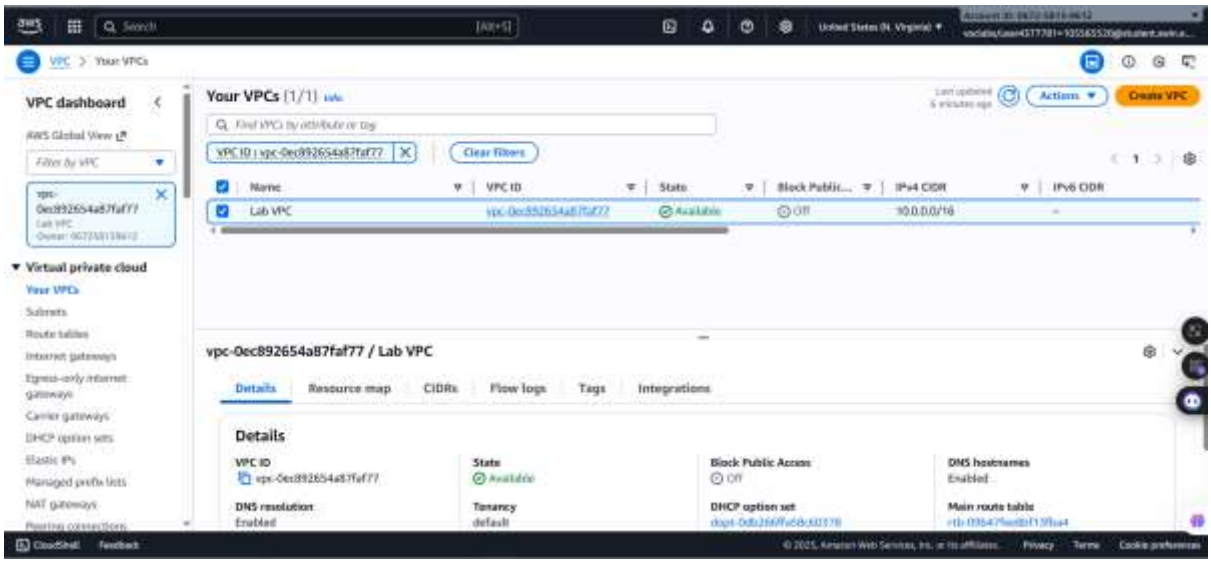


Figure 4: Your VPCs page

Step 1.4: Tick the box next to the **Lab VPC.** Then, in the Details tab, otice that the **IPv4 CIDRs** field has a value of **10.0.0.0/16**, which means that this VPC includes all IP addresses that start with 10.0.x.x.

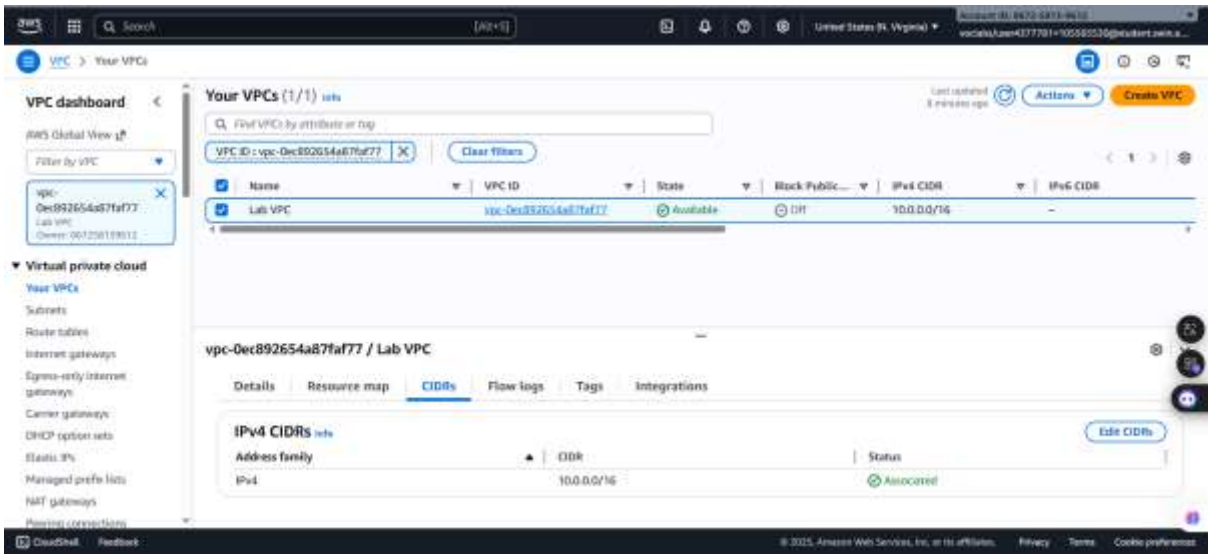


Figure 5: Details of LabVPC

Step 1.5: Select the **Subnets** in the left navigation pane
Here, you can access information about **Public Subnet 1**. In the list of subnets, notice the following:

- The **VPC** column for **Public Subnet 1** shows that this subnet exists inside the **Lab VPC**.
- The **IPv4 CIDR** column has a value of **10.0.0.0/24**, which means that this subnet includes the 256 IP addresses between 10.0.0.0 and 10.0.0.255. Five of these addresses are reserved and unusable.
- The **Availability Zone** column lists the Availability Zone where this subnet resides.

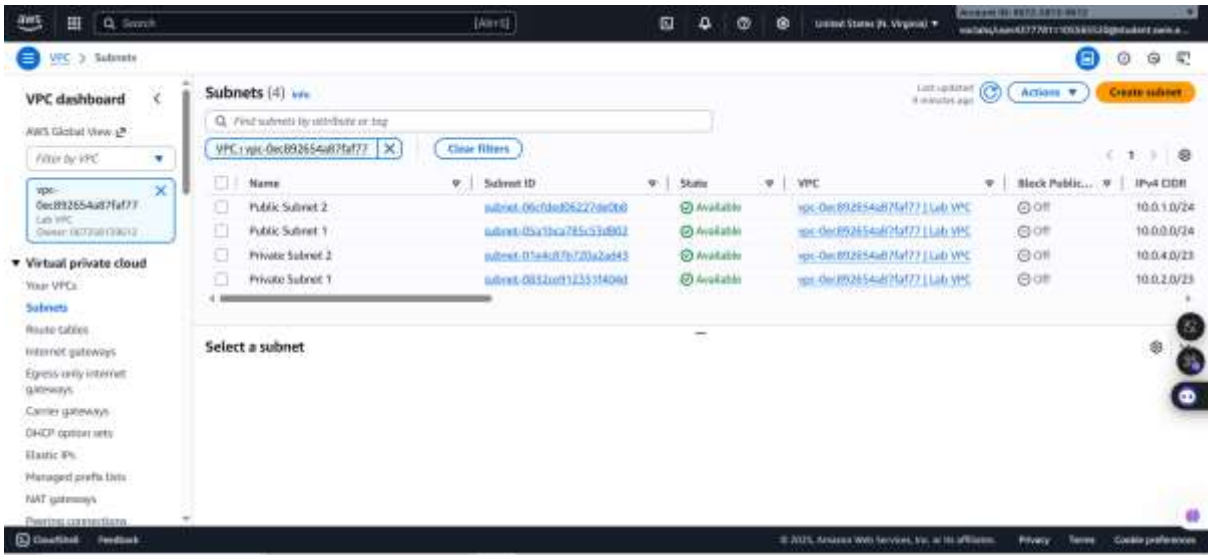


Figure 6: Subnets page

Step 1.6: Tick the box next to the **Public Subnet 1** to reveal more details. After choosing, in the lower half of the page, select the **Route table** tab.
This tab includes details about the routing for this subnet:

- The first entry specifies that traffic destined within the Classless Inter-Domain Routing (CIDR) range for the VPC (**10.0.0.0/16**) is routed within the VPC (**local**).
- The second entry specifies that any traffic destined for the internet (**0.0.0.0/0**) is routed to the internet gateway (**igw-**) that exists in Lab VPC. This setting makes the subnet a *public subnet*.

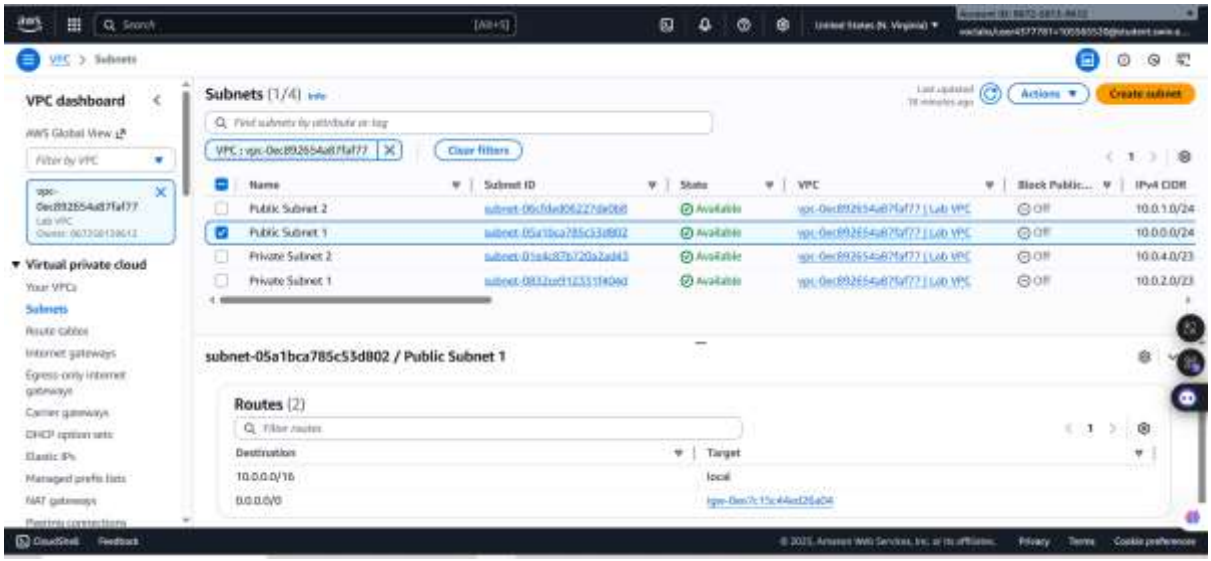


Figure 7: Route table tab details

Step 1.7: Choose the **Network ACL** tab.
This tab has information about the network access control list (network ACL) that is associated with the subnet. The rules are currently set to the default settings. They permit all traffic to flow in and out of the subnet, but you can further restrict the rules by using security groups.

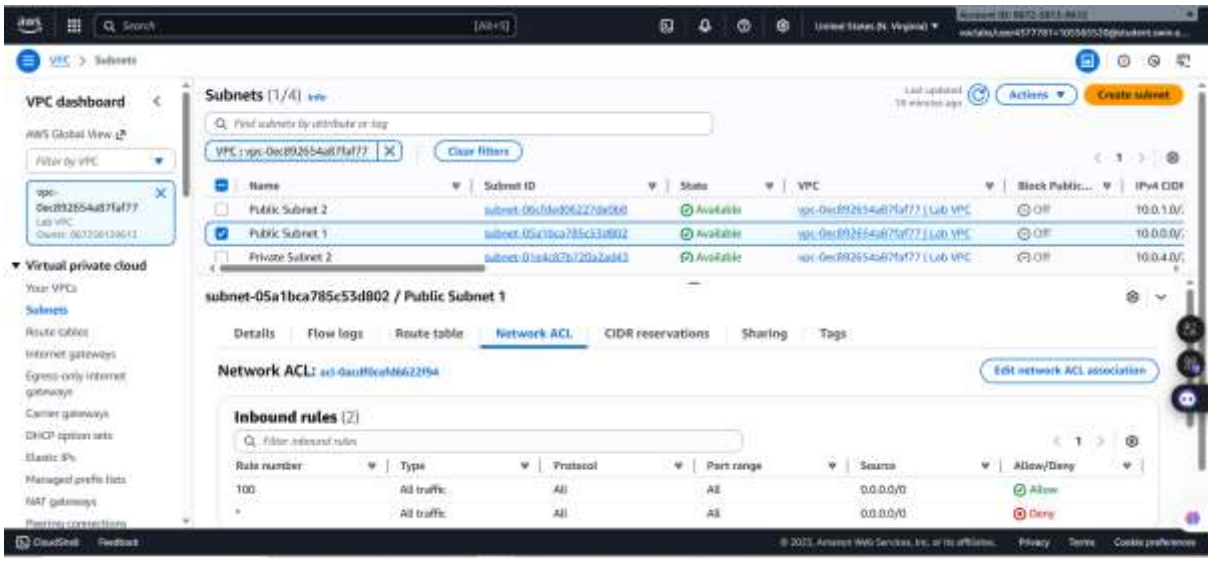


Figure 8: Network ACL tab details

Step 1.8: In the left navigation pane, choose **Internet gateways**.
Notice that an internet gateway with the name **Lab IG** is already attached to the **Lab VPC**.

Figure 8: Network ACL tab details

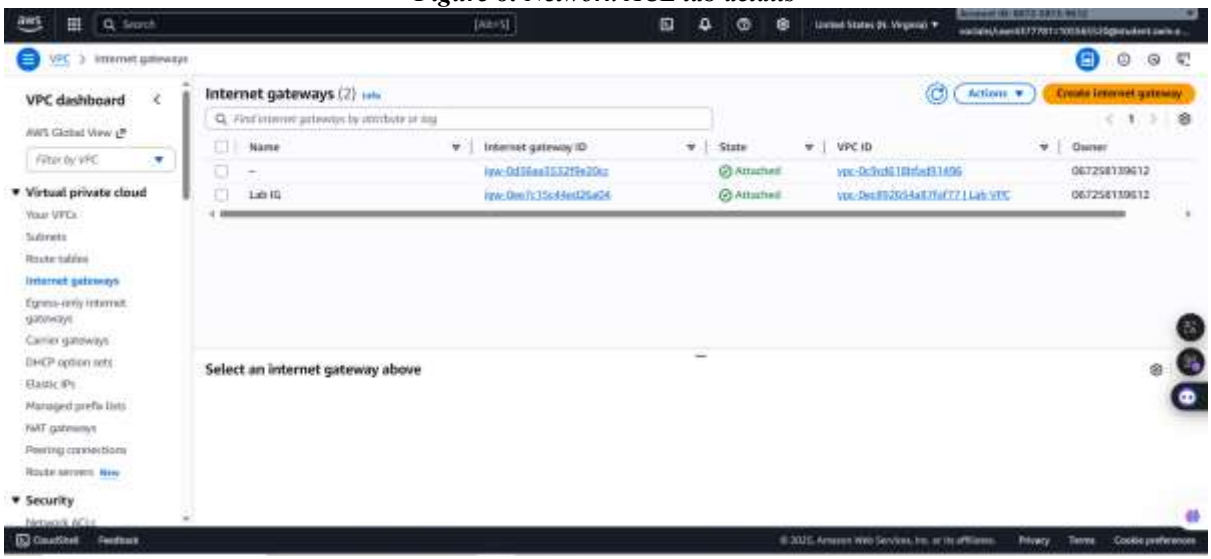


Figure 9: Internet gateways details

Step 1.9: In the left navigation pane, choose **Security group**s and select **Inventory-DB**.
This security group controls incoming traffic to the database.

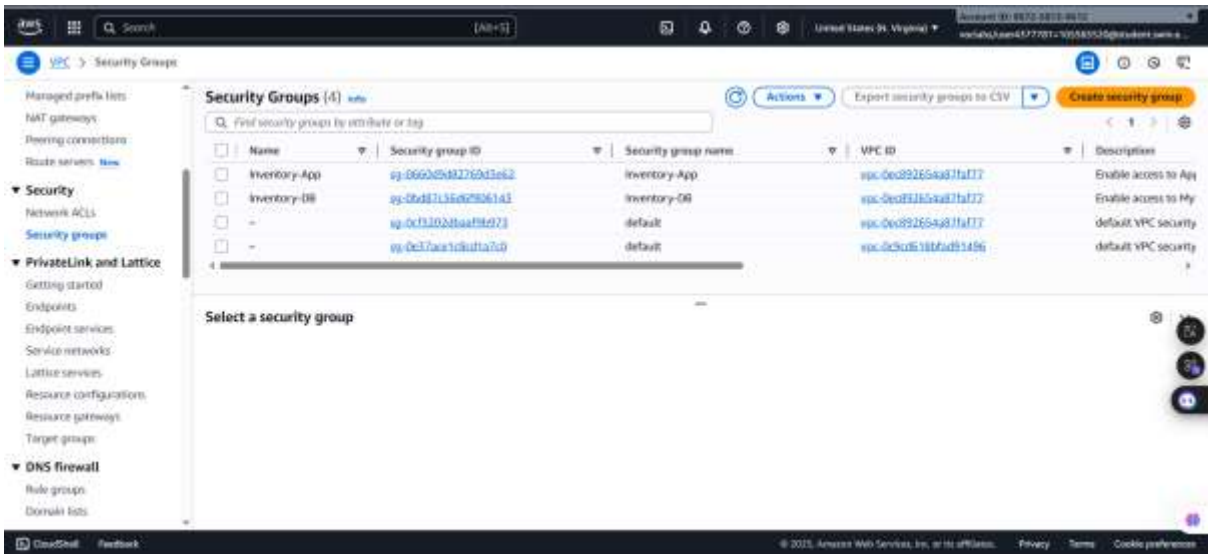


Figure 10: Security groups details

Step 1.10: In the lower half of the page, choose the **Inbound rules** tab.
The rule defined in this security group permits inbound MySQL or Aurora traffic (port 3306) from anywhere in the VPC (10.0.0.0/16). You later modify this setting so that it accepts traffic from only the application servers.

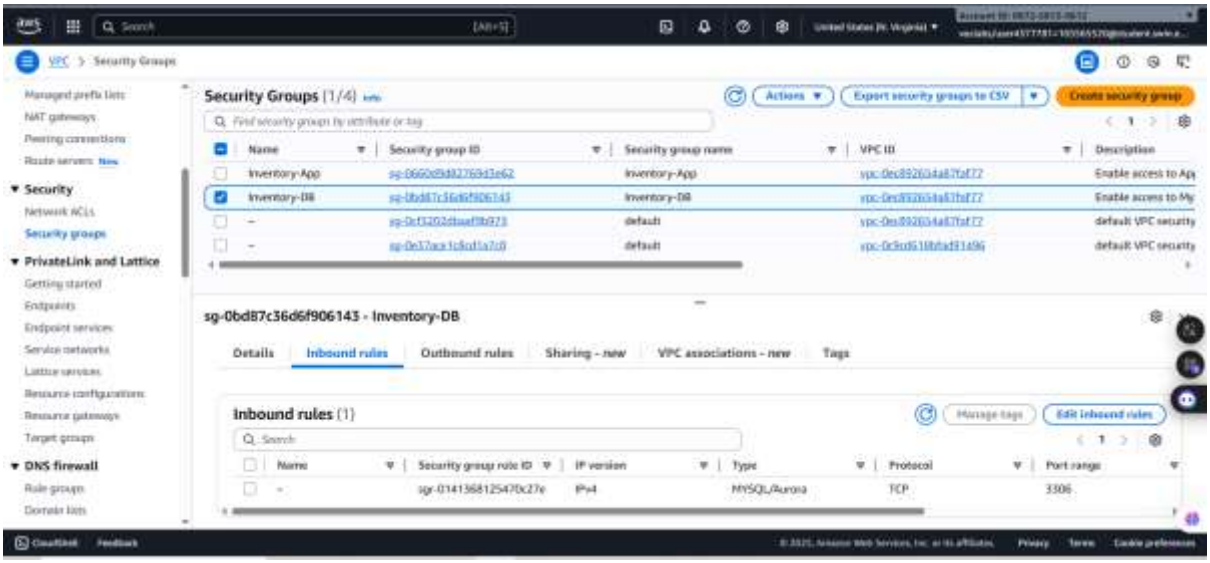


Figure 11: inbound rule details

Step 1.11: Choose the **Outbound rules** tab.
By default, security groups allow all outbound traffic. However, you can modify these settings as needed.

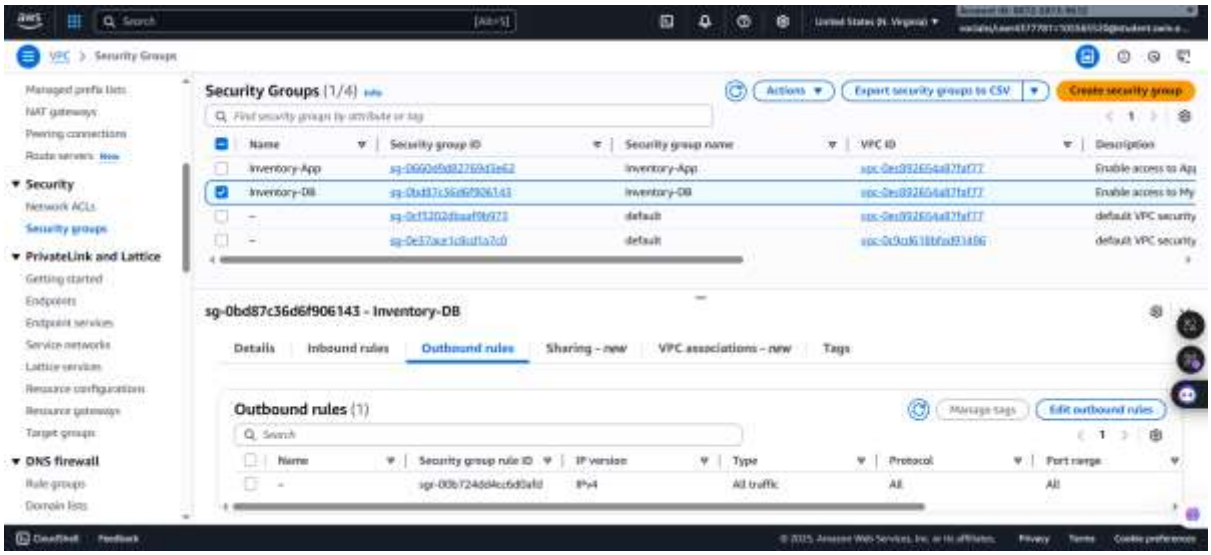
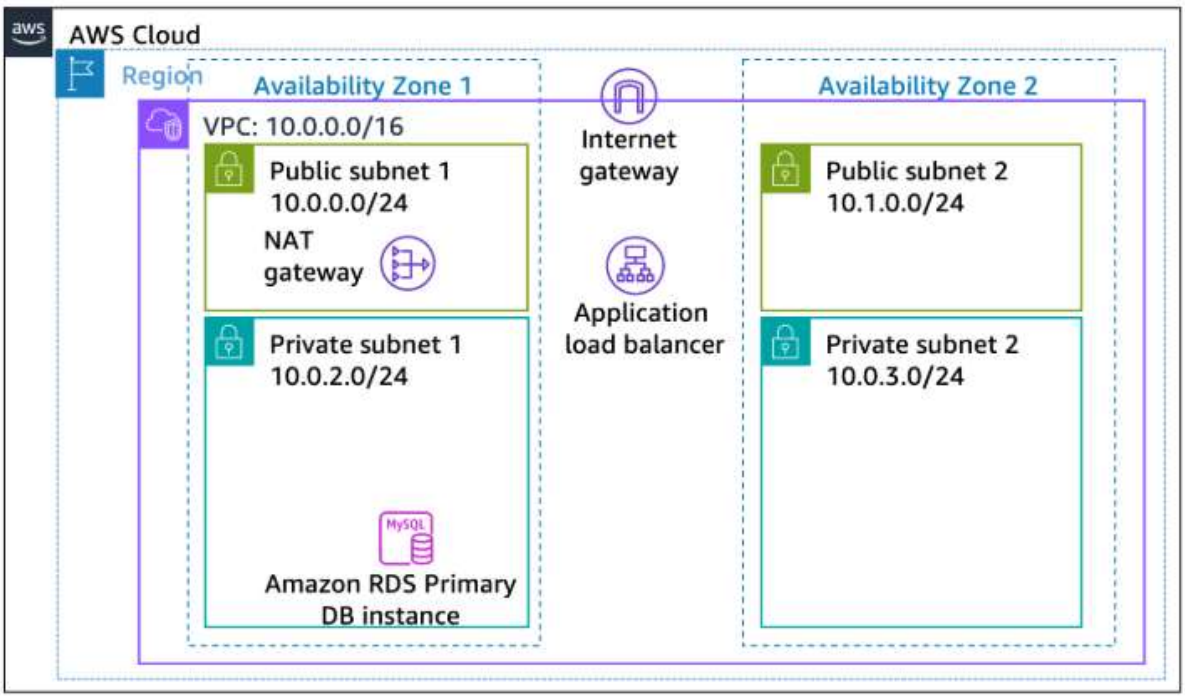


Figure 12: Outboumd rule details

C. Task 2: Creating an Application Load Balancer

To build a highly available application, it is a best practice to launch resources in multiple Availability Zones. Availability Zones are physically separate data centers (or groups of data centers) in the same Region. If you run your applications across multiple Availability Zones, you can provide greater availability if a data center experiences a failure.

Because the application runs on multiple application servers, you need a way to distribute traffic among those servers. You can accomplish this goal by using a *load balancer*. This load balancer also performs health checks on instances and sends requests to only healthy instances.



Step 2.1: On the AWS Management Console, in the search box, enter and choose **EC2** to open the Amazon EC2 console.

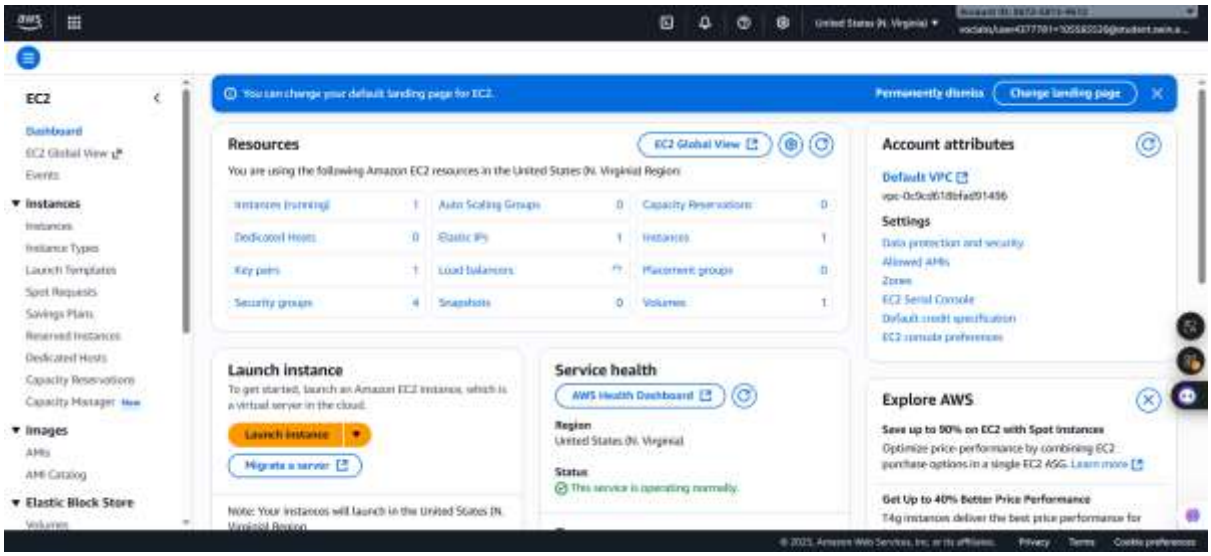


Figure 13: EC2 homepage

Step 2.2: In the left navigation pane, choose **Load Balancers** and then select **Create load balancer** button.

Step 2.3: In the **Load Balancer** types configuration, choose **Create** button in the **Application Load Balancer**

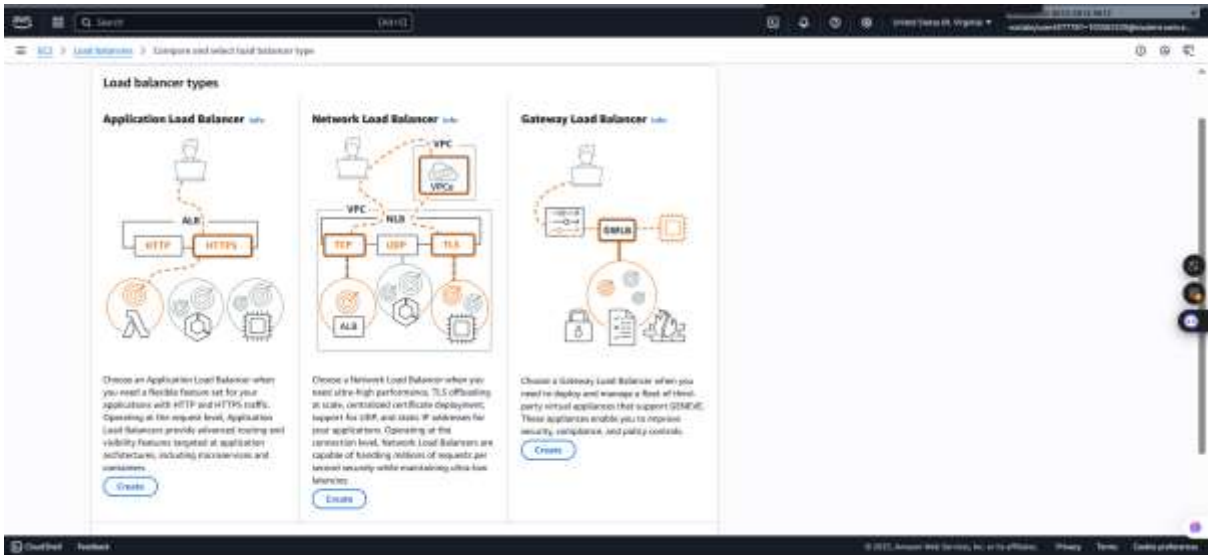


Figure 14: Load Balancer type selection

Step 2.4: In the **Basic configuration** section, for **Load balancer name**, enter *Inventory-LB*.

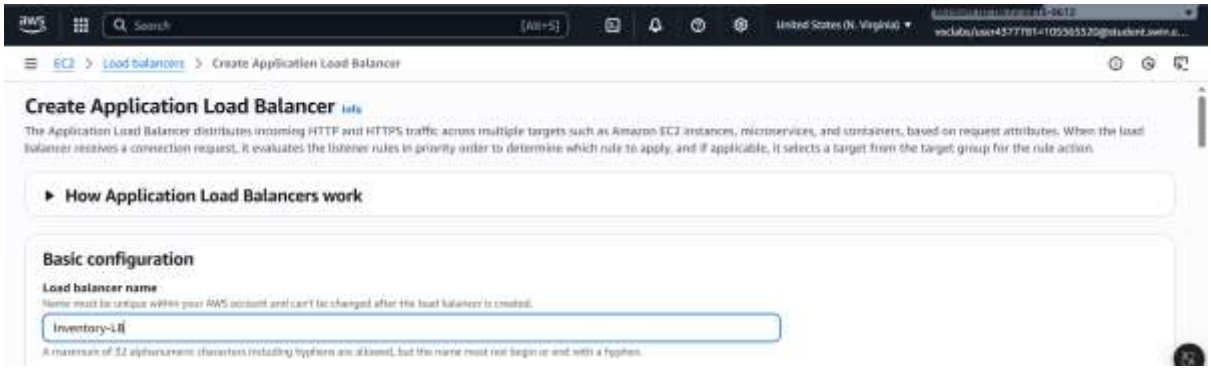


Figure 15: Load Balancer name configuration

Step 2.5: In the **Network mapping** section, configure following settings:

+ **VPC**: Lab VPC

+ **Availability Zones and subnets**: Tick both **us-east-1a** and **us-east-1b** to shouwcase two public subnets

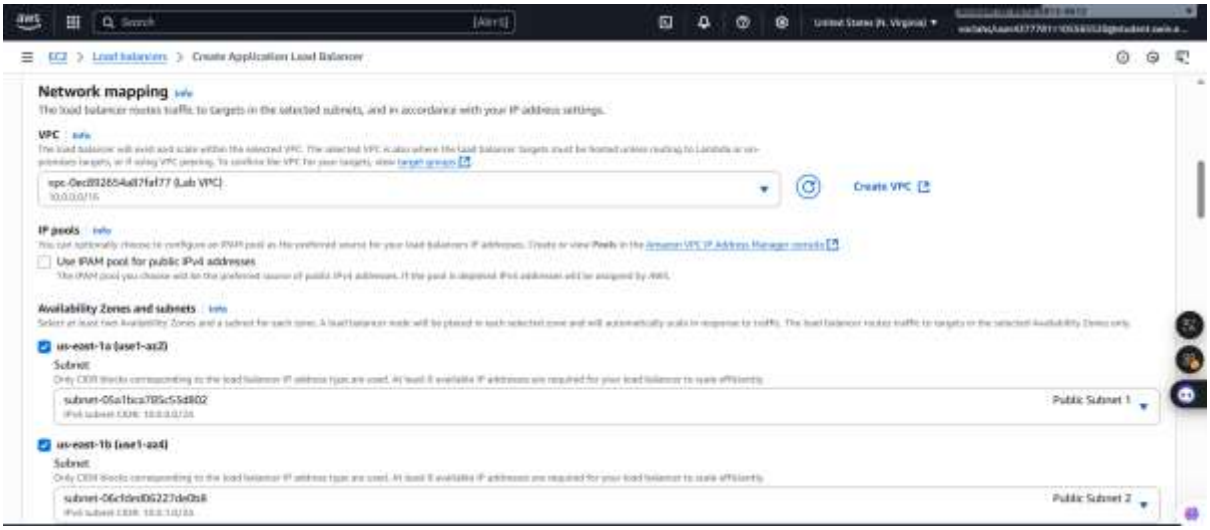


Figure 16: Network mapping configuration

Step 2.6: In the **Security groups** section, select the *create a new security group hyperlink*. This link opens a new browser tab.

Step 2.7: On the **Create security group** page, in the **Basic details** section, configure the following options to create the new security group:

- **Security group name**: Enter Inventory-LB.
- **Description**: Enter *Enable web access to load balancer*.
- **VPC**: From the dropdown list, select **Lab VPC**.

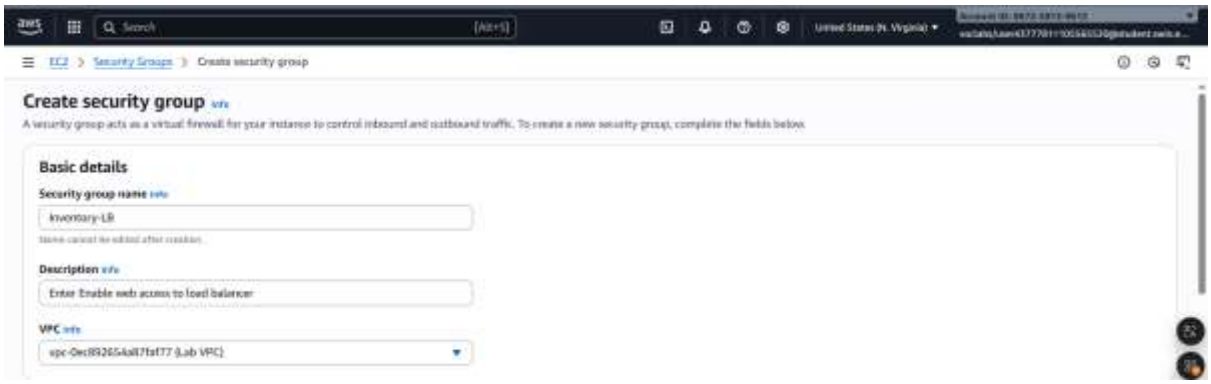


Figure 17: Basic Details configuration

Step 2.8: In the Inbound rules section, choose Add rule, and configure the following options:

- **Type: HTTP**
- **Source: Anywhere-IPv4**

Step 2.9: For **Inbound** rules, choose **Add rule** again, and configure the following options:

- **Type: HTTPS**
- **Source: Anywhere-IPv4**

Figure 18: inbound rule configuration

Step 2.10: After configure above settings are correct, choose **Create security group.**

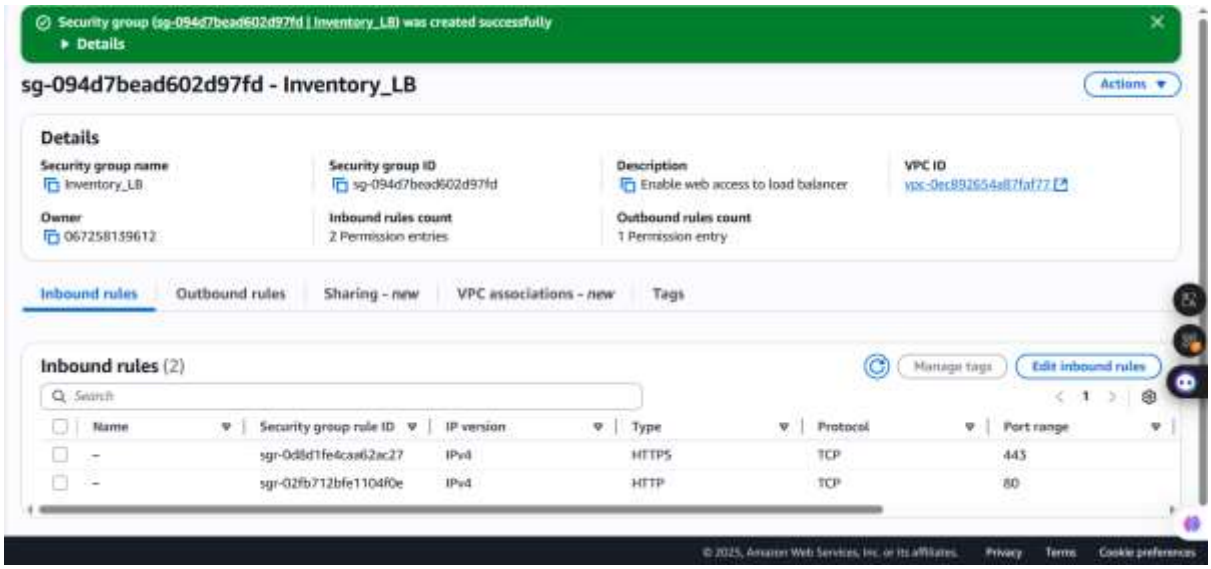


Figure 19: Created successfully

Step 2.11: Return to the **Load Balancer** configuration tab, select the **Inventory-LB** security group that has just created and deselected the default.

Figure 20: Security groups configuratrion

Step 2.12: In the **Listeners and routing** section, choose the **Create target group** link. A new browser tab opens.
Analysis: Target groups define where to send traffic that comes into the load balancer. The Application Load Balancer can send traffic to multiple target groups based on the URL of the incoming request, such as having requests from mobile apps going to a different set of servers. Your web application uses only one target group

Step 2.13: For **Step 1: Specify group** details, configure the following options:

- In the **Basic configuration** section, configure the following options:
  - **Choose a target type**: Choose **Instances**.

Figure 21: Target type configuratrion

  - **Target group name**: Enter Inventory-App.
  - **VPC**: Ensure that **Lab VPC** is chosen.

Figure 22: Target group name and VPC configuratrion

- In the **Health checks** section, expand Advanced health check settings, and configure the following options:
  Note: The Application Load Balancer automatically performs health checks on all instances to ensure that they are responding to requests. The default settings are recommended, but you make them slightly faster for use in this lab.
  - **Healthy threshold**: Enter 2.
  - **Interval:** Enter 10 (seconds).

The configurations you have chosen results in the health check being performed every 10 seconds. If the instance responds correctly twice in a row, it is considered healthy.

Figure 23: Healthcheck configuratrion

Step 2.14: Choose **Next.**
The **Step 2: Register targets** screen appears.
Note: Targets are the individual instances that respond to requests from the load balancer. You do not have any web application instances yet, so you can skip this step.

Step 2.15: Review the settings, and choose **Create target group.**

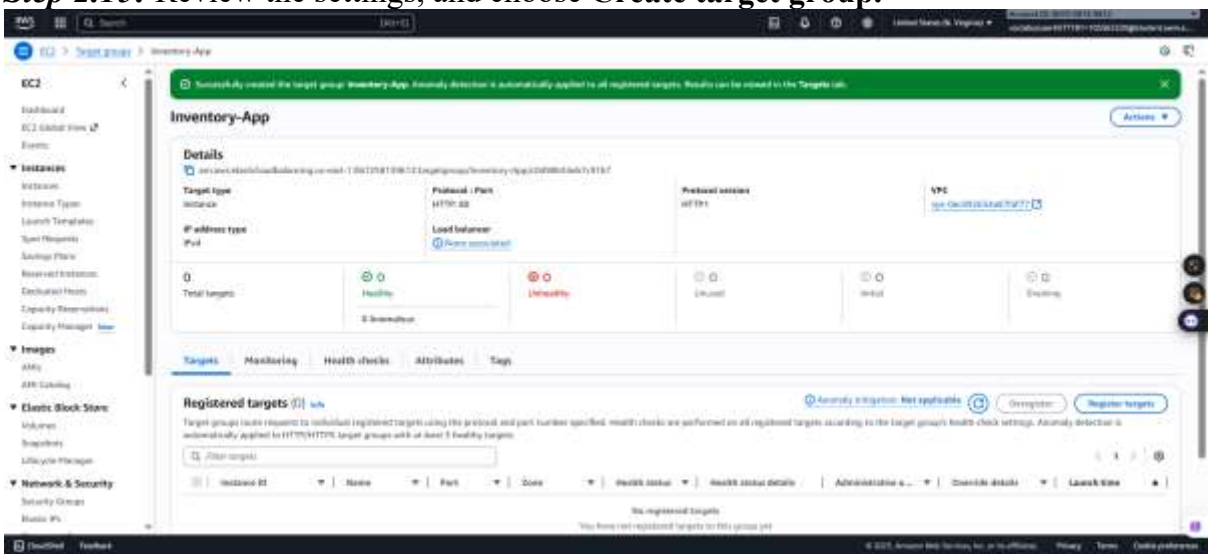


Figure 24: Create target group successful

Step 2.16: Return to the browser tab where you already started defining the load balancer.

Step 2.17: In the Listeners and routing section. From the **Default action** dropdown list, choose the **Inventory-App** target group that just created.

Figure 25: Ttarget group selection

Step 2.18: Scroll to the bottom of the page, and choose **Create load balancer** and choose **View load balancer.**

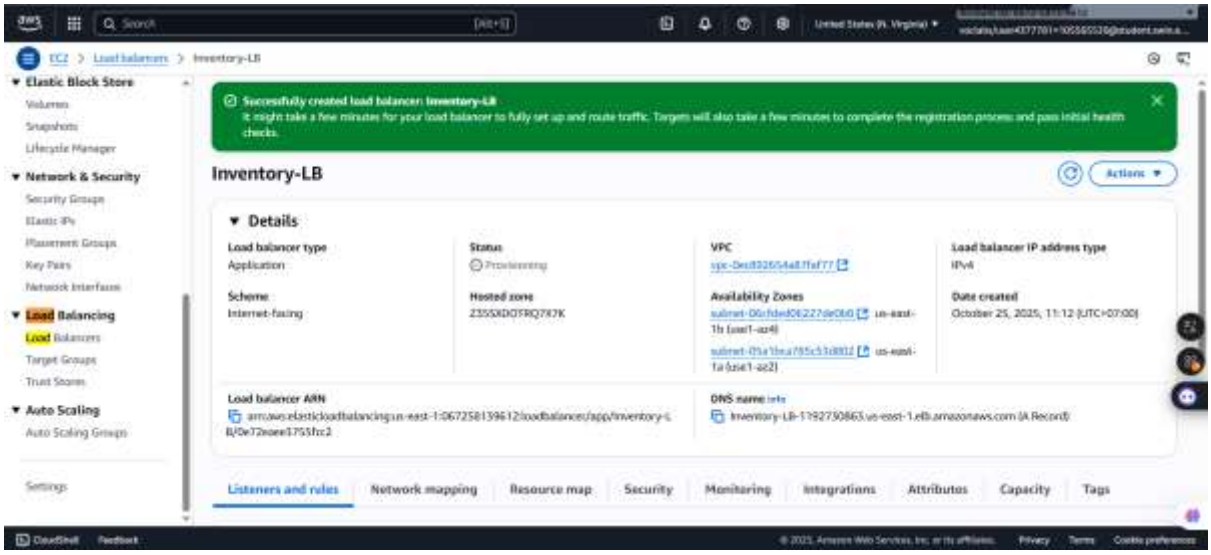
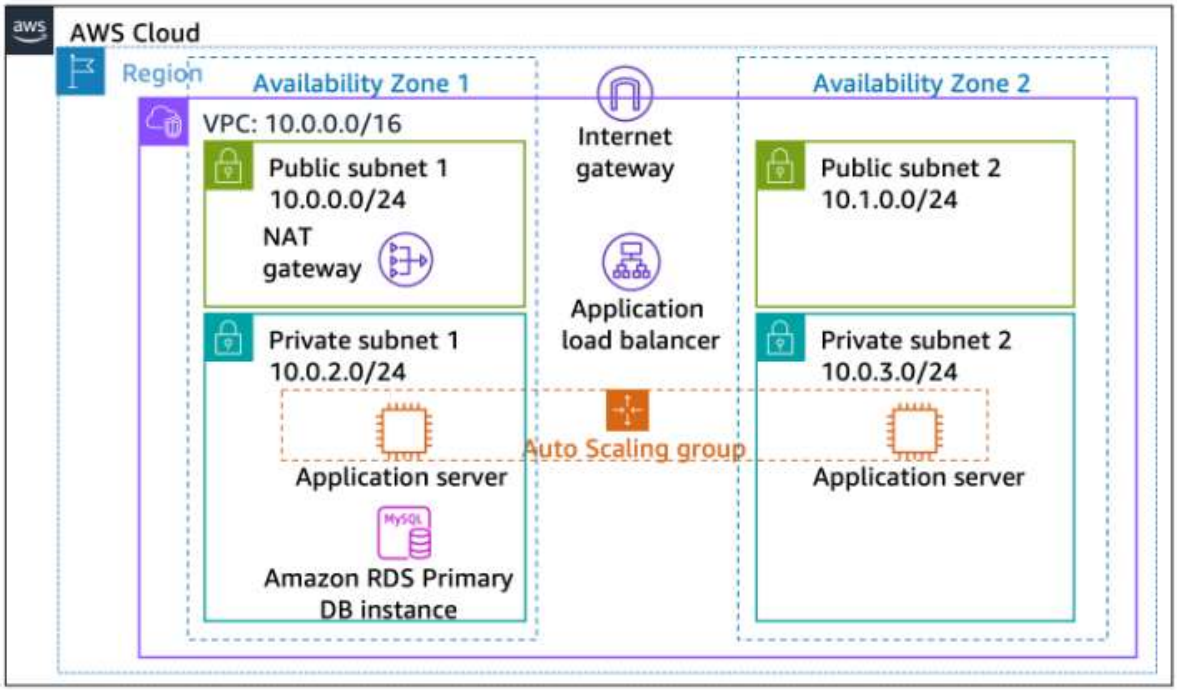


Figure 26: Create successful

D. Task 3: Creating an Auto Scaling group



I. Task 3.1: Creating an AMI for Auto Scaling

Step 3.1.1: Search, select the **EC2,** and then choose **Instances** in the left navigation pane

Step 3.1.2: Observe the Status Check of the **Web Server 1** displays **2/2 checks passed.**

Step 3.1.3: Select **Web Server 1**, from the **Actions** dropdown list, choose **Image and templates > Create image.**

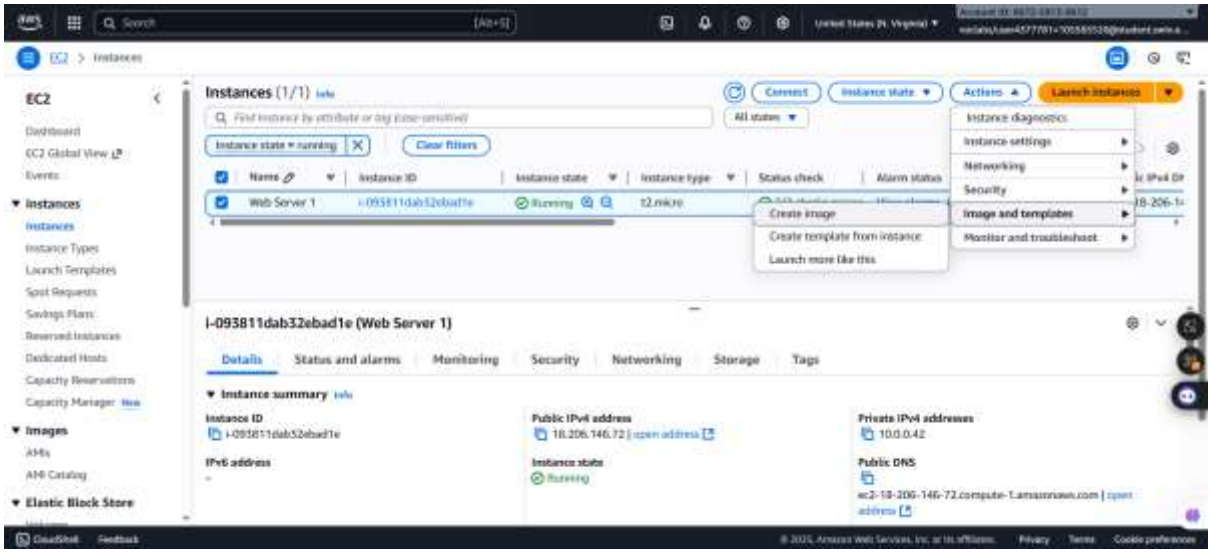


Figure 27: Actions selection

Step 3.1.4: On the **Create image** page, configure the following options:

- **Image name**: Enter *Web Server AMI.*
- **Image description:** Enter *Lab AMI for Web Server.*

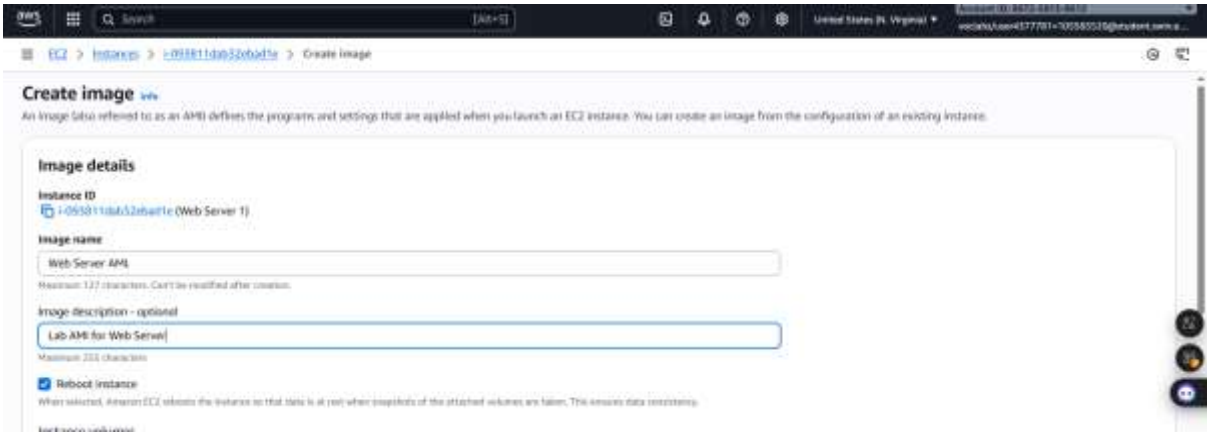


Figure 28: Image details configuration

Step 3.1.5: Choose **Create image**.

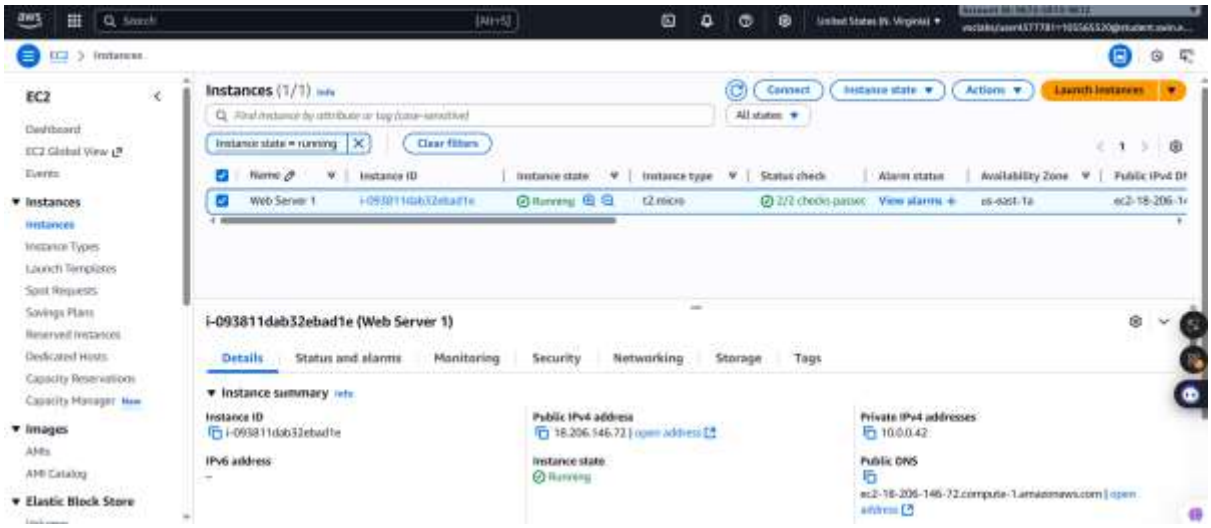


Figure 29: Create image successfully

II. Task 3.2: Creating a launch template and an Auto Scaling group

Step 3.2.1: In the left navigation pane, choose **Launch Templates.**

Step 3.2.2: Configure following settings:

- In the **Launch template name and description** section, configure the following options:

- For **Launch template** name, enter *Inventory-LT*.
- For **Auto Scaling guidance**, select **Provide guidance to help me set up a template that I can use with EC2 Auto Scaling.**

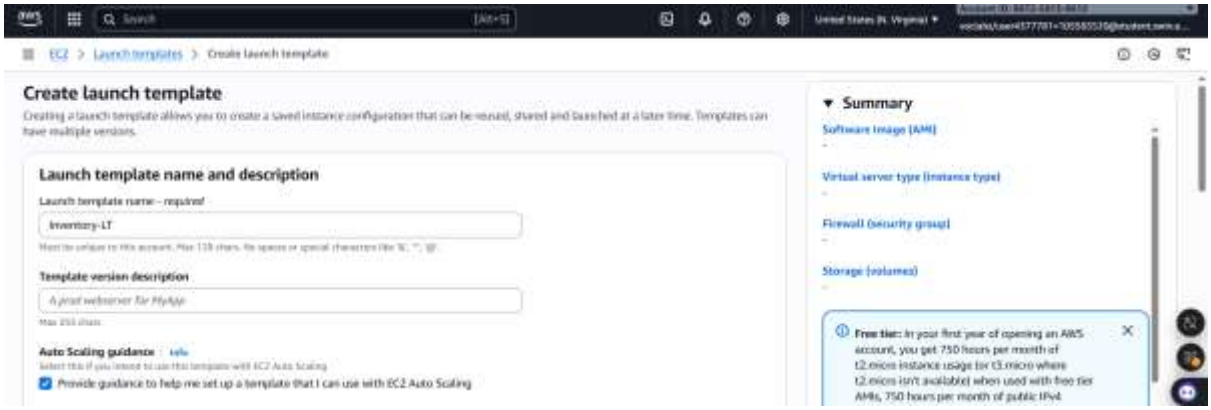


Figure 30: Launch template name and description configuration

- In the **Application and OS Images (Amazon Machine Image)** section, configure the following options:

- Choose **My AMIs.**
- For **Amazon Machine Image (AMI)**, choose **Web Server AMI.**

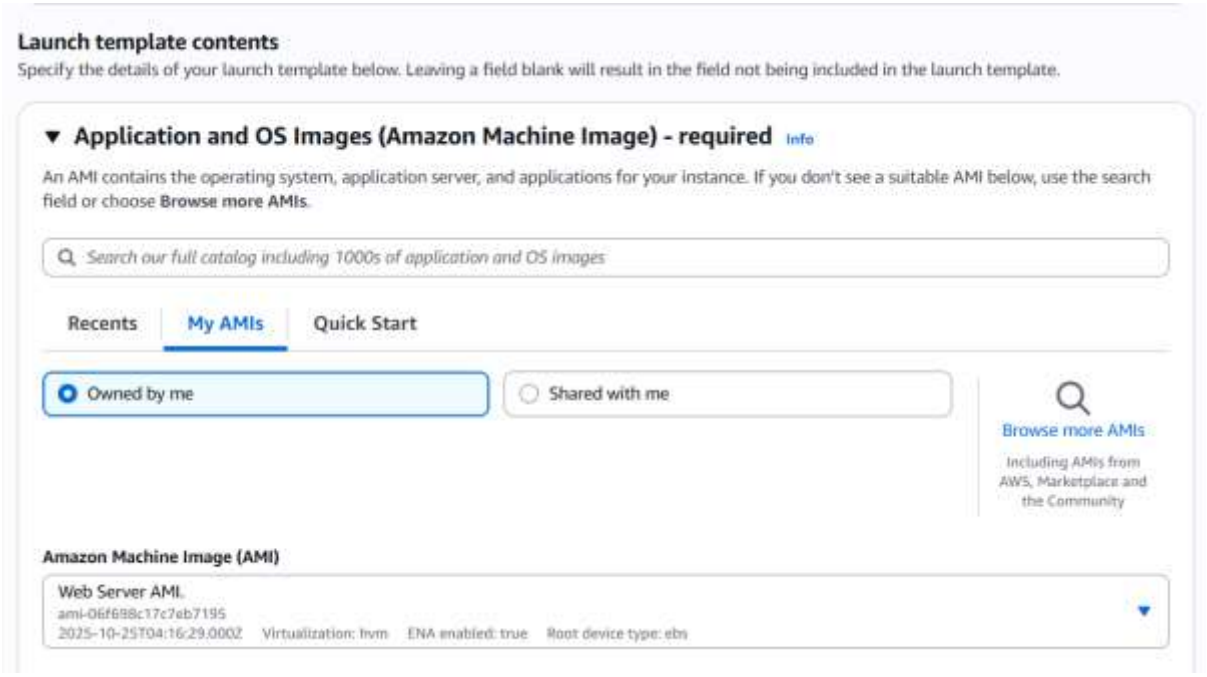


Figure 31: Application and OS Images (Amazon Machine Image) configuration

- In the Instance type section, choose **t2.micro.**

Figure 32: Instance type configuration

- In the **Key pair (login)** section, for **Key pair name**, choose **vockey.**

Figure 33: Key pair configuration

- In the **Network settings** section, configure the following options:
  - For **Firewall (security groups),** choose **Select existing security group**.
  - For **Security groups**, choose **Inventory-App.**

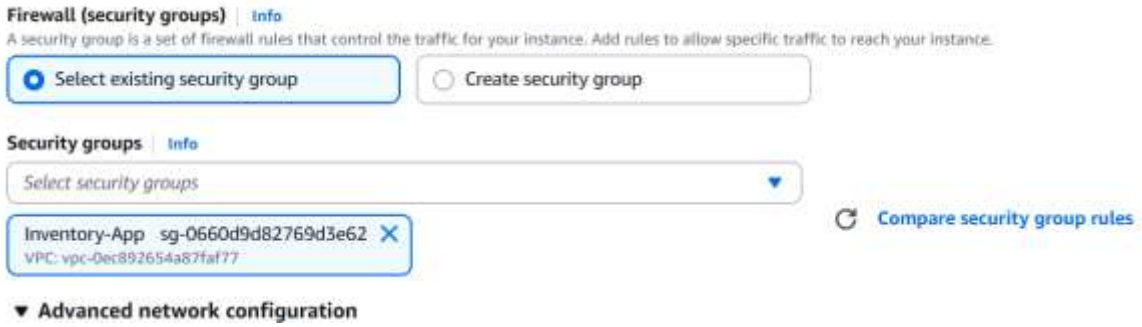


Figure 34: Firewall configuration

- Expand the **Advanced details** section, and configure the following options:
  - For **IAM instance profile**, choose **Inventory-App-Role**.

Figure 35: IAM instance profile configuration

  - For **Detailed CloudWatch monitoring**, choose **Enable**. **Note:** This option allows Auto Scaling to react quickly to changing utilization.

Figure 36: Detailed Cloudwatch monitoring configuration

  - In the **User data** box, enter the following script:

```
#!/bin/bash
# Install Apache Web Server and PHP
yum install -y httpd mysql
amazon-linux-extras install -y php7.2
# Download Lab files
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACACAD-3-113230/12-lab-mod10-
guided-Scaling/s3/scripts/inventory-app.zip
unzip inventory-app.zip -d /var/www/html/
# Download and install the AWS SDK for PHP
wget https://github.com/aws/aws-sdk-php/releases/download/3.62.3/aws.zip
unzip aws -d /var/www/html
# Turn on web server
chkconfig httpd on
service httpd start
```

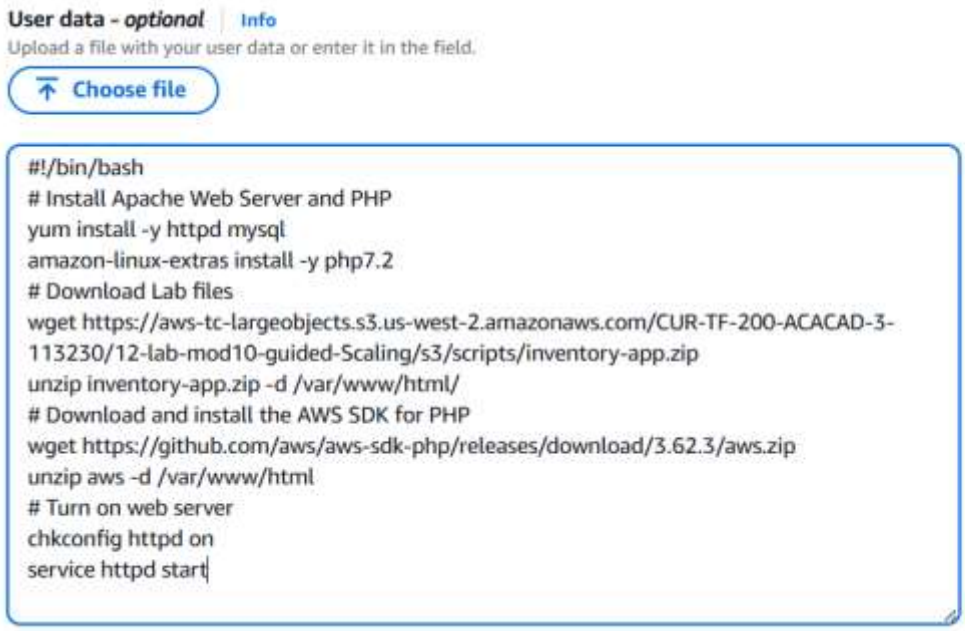


Figure 37: User Data configuration

Step 3.2.3: Choose **Create launch template.**

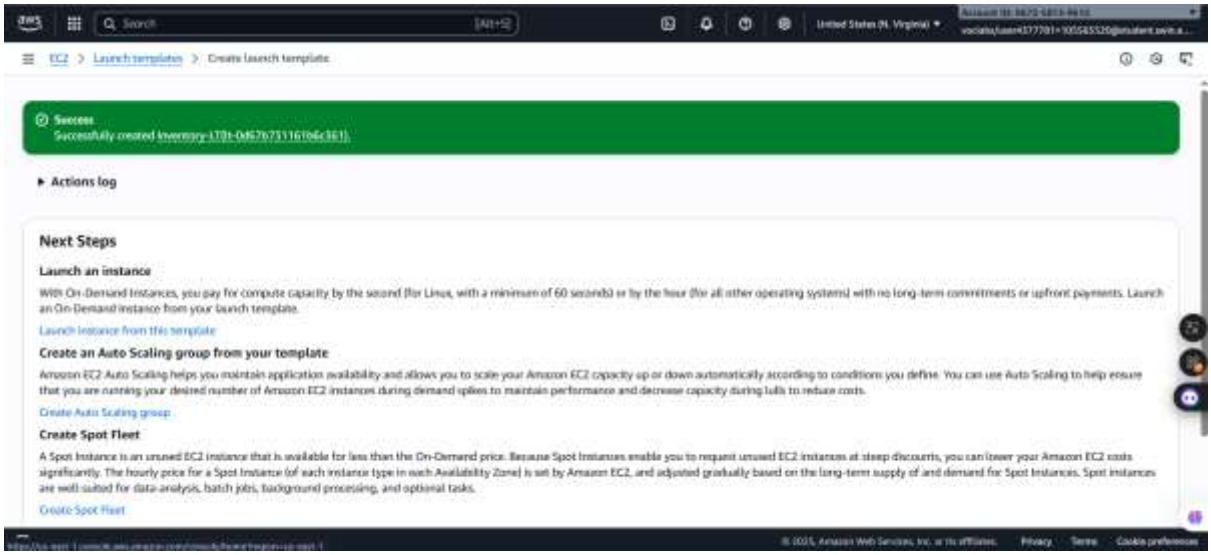


Figure 38: Create successfully

Step 3.2.4: After creating successful, choose the link for the **Inventory-LT** launch template. From the **Actions** dropdown list, choose **Create Auto Scaling group**.

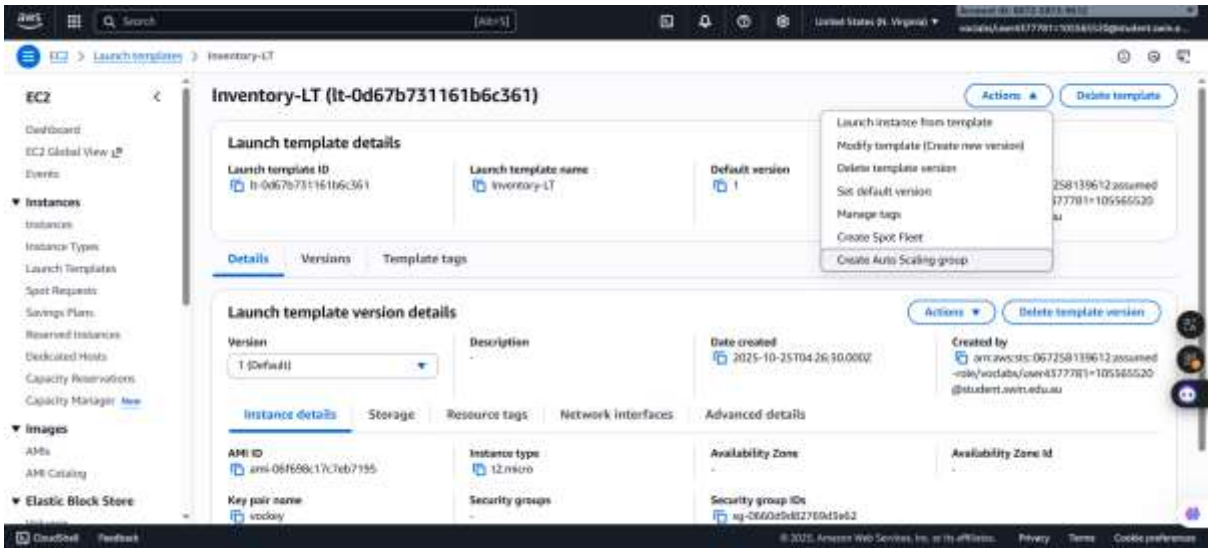


Figure 39: Actions seletcion

Step 3.2.5: For Step 1: Choose launch template or configuration, configure the following options:

- **Auto Scaling group name**: Enter *Inventory-ASG* (ASG stands for_Auto Scaling group).
- **Launch template**: Confirm that the Inventory-LT template that you just created is selected.

After that, choose **Next**

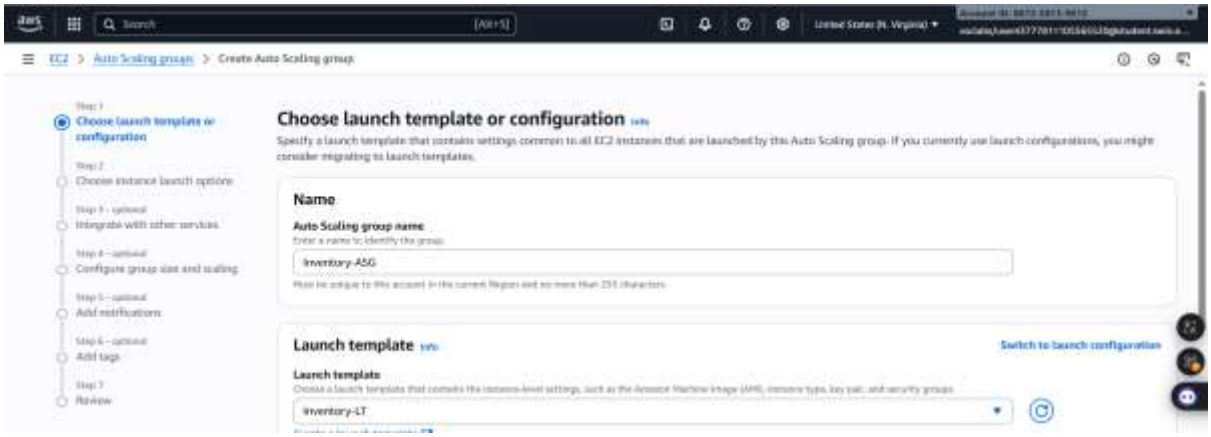


Figure 40: Step 1 configuration

Step 3.2.6: For **Step 2: Choose instance launch options**, configure the following options:

- **VPC**: Choose **Lab VPC.**
- **Availability Zones and subnets**: Choose **Private Subnet 1**, and then choose **Private Subnet 2.**

These settings launch EC2 instances in private subnets across both Availability Zones.

After that, choose **Next**

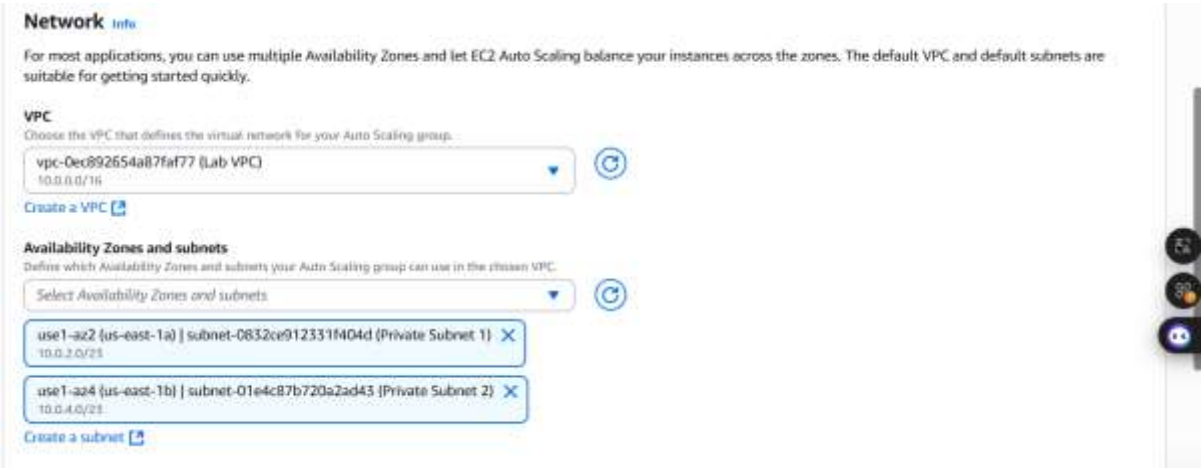


Figure 41: Step 2 configuration

Step 3.2.7: For **Step 3: Configure advanced options**, configure the following options:

- In the **Load balancing** section, configure the following options:
  - Choose **Attach to an existing load balancer.**
  - From the **Existing load balancer target groups** dropdown list, choose **Inventory-App.**

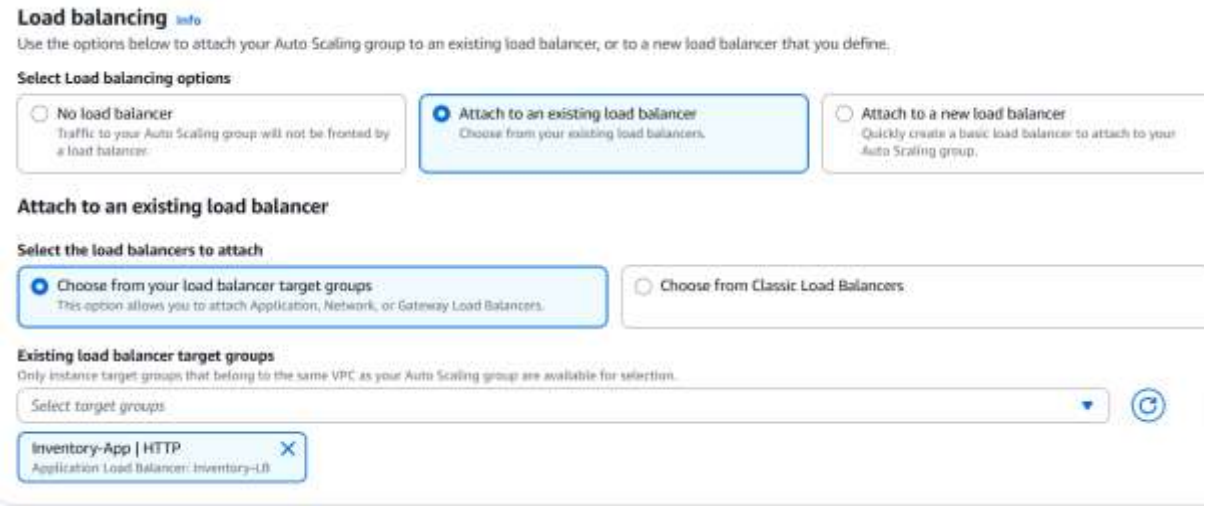


Figure 42: Load Balancing configuration

- In the **Health checks** section, configure the following options:
  - Select **Turn on Elastic Load Balancing health checks.**
  - For **Health check grace** period, enter **90** seconds.

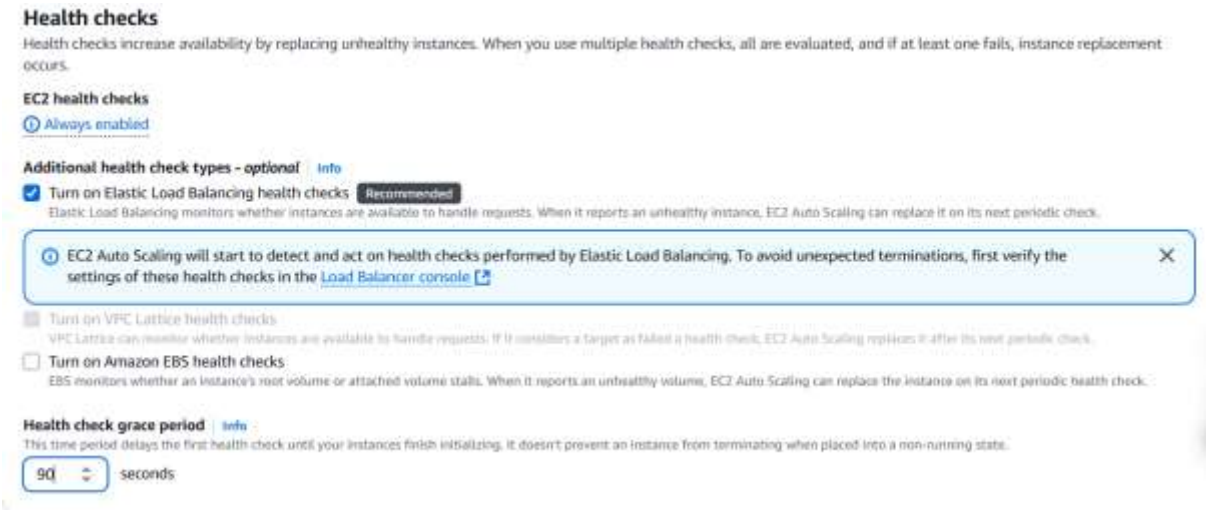


Figure 43: Health checks configuration

- In the Additional settings section, select Enable group metrics collection within CloudWatch.

Figure 44: Additional settings configuration

This setting captures metrics at 1-minute intervals, which allows Auto Scaling to react quickly to changing usage patterns.
After that, choose **Next**

Step 3.2.8: For **Step 4: Configure group size and scaling policies**, configure the following options:

- For **Group size**, for **Desired capacity**, enter 2.
- For **Scaling**, configure the following options:
  - For **Min desired capacity**, enter 2.
  - For **Max desired capacity**, enter 2.
    These settings allow Auto Scaling to automatically add or remove instances, always keeping 2–6 instances running.
- For Automatic scaling - optional, choose No scaling policies.
  For this lab, you maintain two instances at all times to ensure high availability. If the application is expected to receive varying loads of traffic, you can also create scaling policies that define when to launch or terminate instances. However, you do not need to create scaling policies for the inventory application in this lab.

After that, choose **Next**

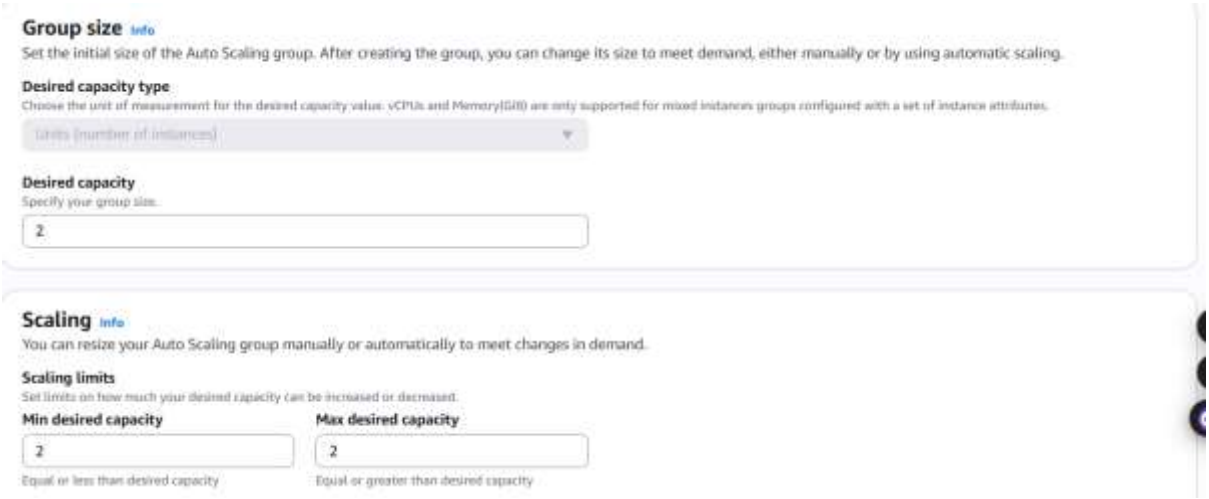


Figure 45: Group Size configuration

Step 3.2.9: For **Step 5: Add notifications**, you do not need to configure any settings. Choose **Next.**

Step 3.2.10: For **Step 6: Add tags**, choose **Add tag,** and configure the following options:

- **Key:** Enter Name.
- **Value:** Enter Inventory-App.

These settings tag the Auto Scaling group with a name, which also appears on the EC2 instances that are launched by the Auto Scaling group. You can use tags to identify which EC2 instances are associated with which application. You could also add tags such as Cost Center to assign application costs in the billing files.
After that, choose **Next**

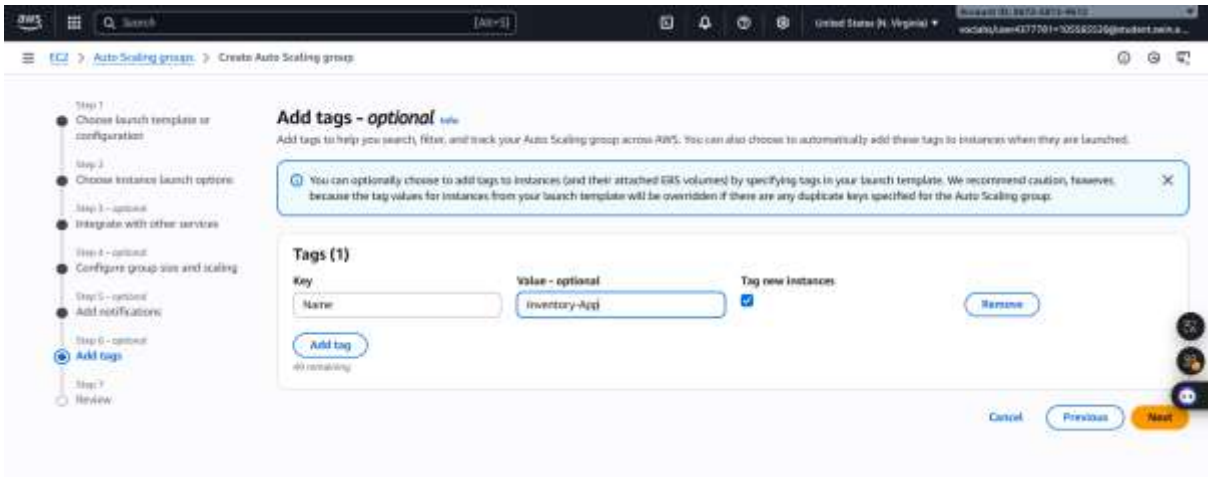


Figure 46: Tag configuration

Step 3.2.11: For **Step 7: Review**, review the details of your Auto Scaling group, and then choose **Create Auto Scaling group**.

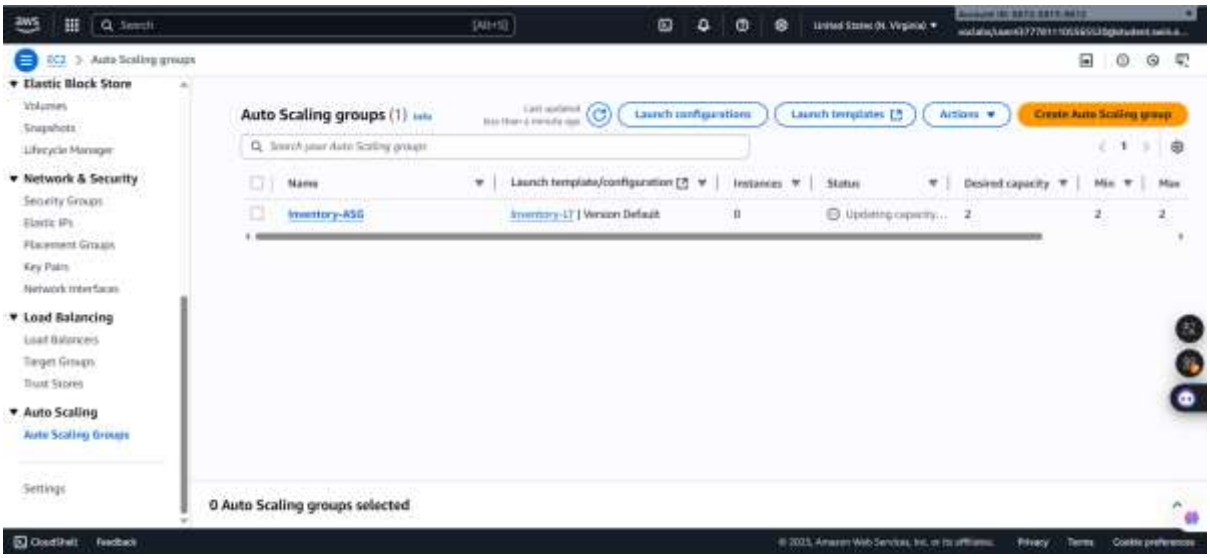
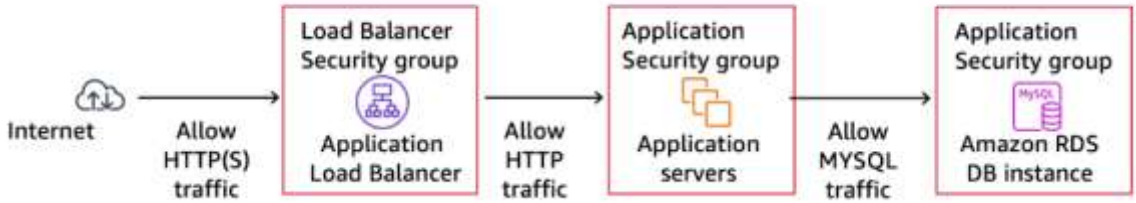


Figure 47: Create successfully

E. Task 4: Updating security groups



I. Task 4.1: Creating an AMI for Auto Scaling

II. Task 4.2: Configuring the application security group

Step 4.2.1: In the left navigation pane, choose **Security Groups** and select **Inventory-App.**

Step 4.2.2: Choose the **Inbound rules** tab.
The security group is currently empty. You now add a rule to accept incoming HTTP traffic from the load balancer. You do not need to configure HTTPS traffic because the load balancer was configured to forward HTTPS requests through HTTP. This practice offloads security to the load balancer, reducing the amount of work that is required by the individual application servers.

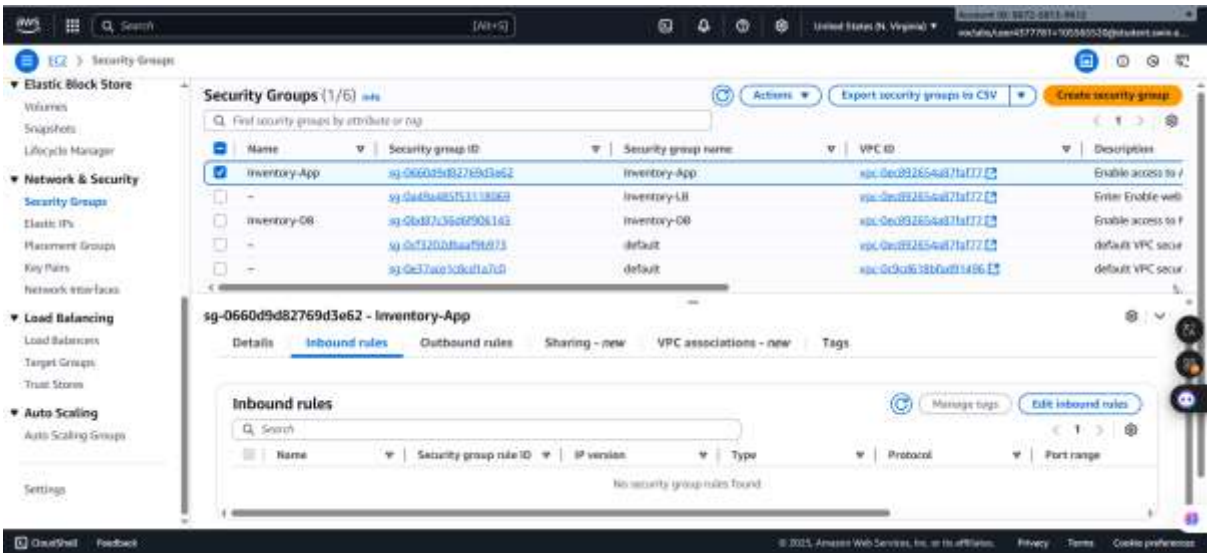


Figure 48: Inventory-App selected

Step 4.2.4: Choose **Edit inbound rules** with following settings:

- For **Type**, choose **HTTP**.
- For **Port**, enter 80.
- For **Source**, configure the following options:
  - Choose the search box to the right of **Custom**.
  - Delete the current contents.
  - Enter **sg**.
  - From the list that appears, select **Inventory-LB**.
  - For **Description**, enter *Traffic from load balancer*.

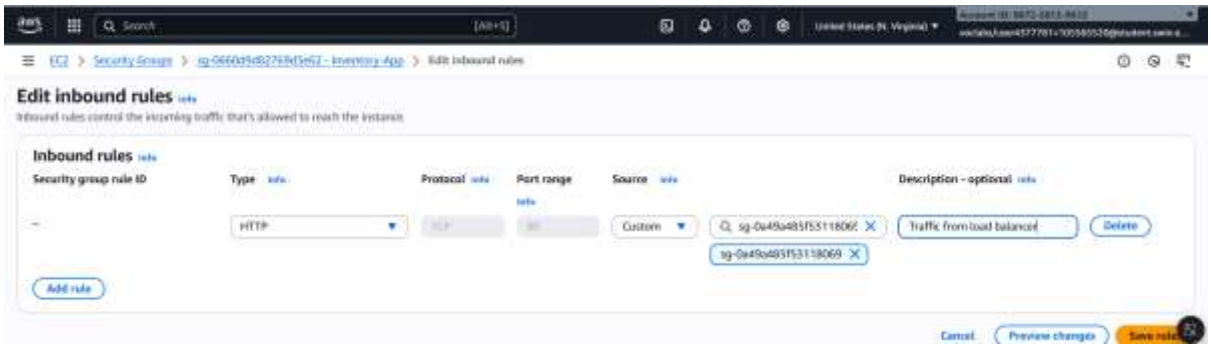


Figure 49: Inbound rule configuration

Step 4.2.5: Choose **Save rules.**

III. Task 4.3: Configuring the database security group

Step 4.3.1: In the **Security groups** list, select **Inventory-DB** and make sure that no other security groups are selected.

Step 4.3.2: In the **Inbound rules** tab, choose **Edit inbound** rules, and configure the following options:

- For the existing rule, choose **Delete.**

- Choose **Add rule.**
- For **Type**, choose **MYSQL/Aurora.**
- For **Source**, configure the following options*:*
  - Choose the search box to the right of **Custom**.
  - Enter **sg.**
  - From the dropdown list, select **Inventory-App.**
  - For **Description**, enter *Traffic from application servers*.

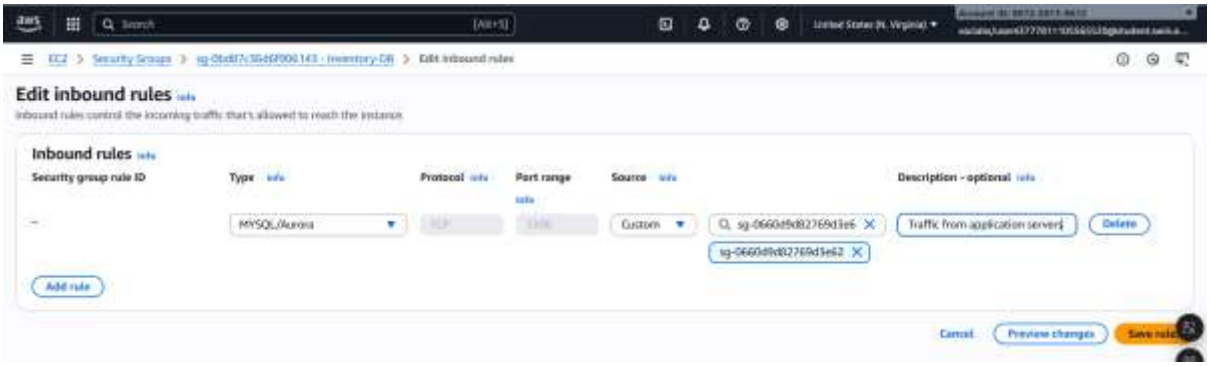


Figure 50: Outbound rule configuration

Step 4.3.3: Choose **Save rules.**

F. Task 5: Testing the application

Your application is now ready for testing.

In this task, you confirm that your web application is running. You also test that it is highly available.

Step 5.1: In the left navigation pane, choose **Target Groups** and select **Inventory-App**.

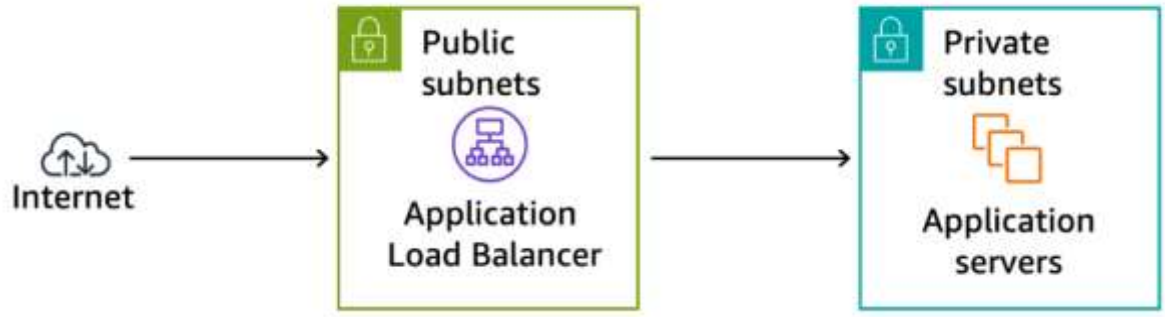
Step 5.2: Choose the **Targets tab.**
This tab should show two **Registered targets**. The **Health status** column shows the results of the load balancer health check that is performed against the instances.

Step 5.3: In the **Registered targets** area, choose the refresh icon until the **Status** for both instances appears as healthy.
If the status does not eventually change to healthy, ask your educator for help with diagnosing the configuration.
You test the application by connecting to the load balancer, which then sends your request to one of the EC2 instances. You first need to retrieve the DNS name of the load balancer.

Step 5.4: In the left navigation pane, choose **Load Balancers**, and then choose **Inventory-LB.**

Step 5.5: In the **Details** tab in the lower half of the window, copy the **DNS name** to your clipboard.

It should be similar to **Inventory-LB-xxxx.elb.amazonaws.com**.



G. Task 6: Testing high availability

Your application is configured to be highly available. You can prove the application's high availability by terminating one of the EC2 instances.

Step 6.1: Return to the **EC2** console tab in your web browser and do not close the web application tab

Step 6.2: In the left navigation pane, choose **Instances.**

Step 6.3: Select one of the **Inventory-App** instances. It does not matter which one you select.

Step 6.4: Choose **Instance state > Terminate instance**.

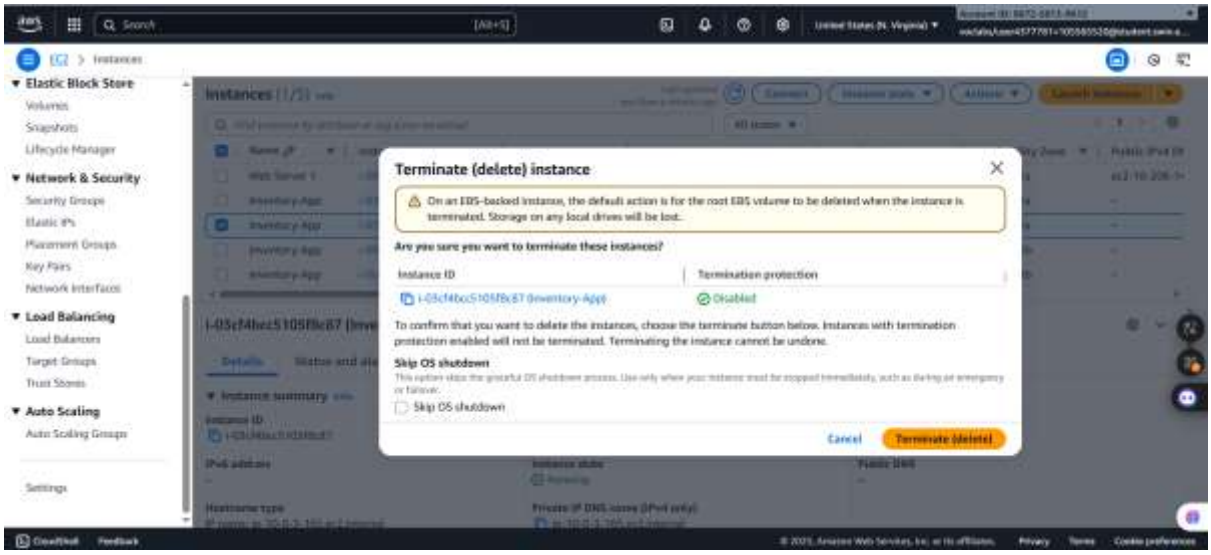


Figure 51: Announcement before terminate

Step 6.5: In the **Terminate instance?** window, choose **Terminate**.
In a short time, the load balancer health checks notice that the instance is not responding. The load balancer automatically routes all requests to the remaining instance.

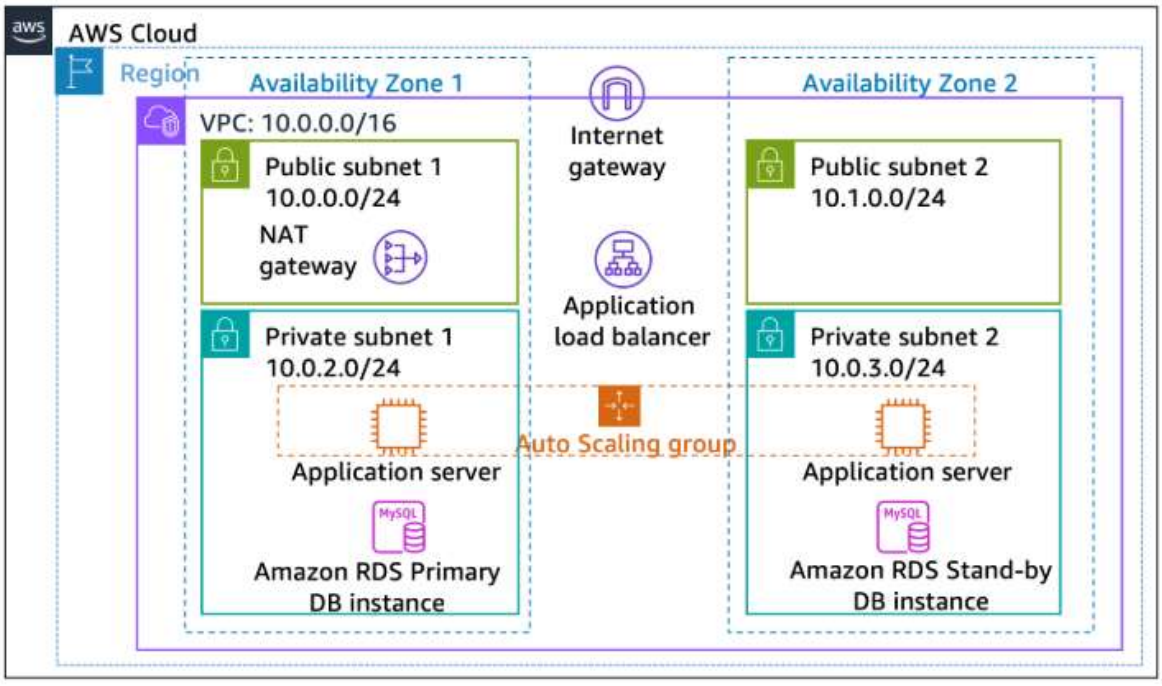
Step 6.6: Return to the web application tab in your web browser, and reload the page several times.
You should notice that the **Availability Zone** that is shown at the bottom of the page stays the same. Although an instance failed, your application remains available.
After a few minutes, Amazon EC2 Auto Scaling also notices the instance failure. It is configured to keep two instances running, so Amazon EC2 Auto Scaling automatically launches a replacement instance.

Step 6.7: Return to the Amazon EC2 console tab where you have the instances list displayed. In the upper-right area, choose the refresh icon every 30 seconds until a new EC2 instance appears.
After a few minutes, the health check for the new instance should become healthy. The load balancer resumes sending traffic between the two Availability Zones. You can reload your web application tab to see this happen.
This task demonstrates that your application is now highly available.

H. Task 7: Making the database highly available (Optional)

This task is optional. You can work on this task if you have remaining lab time.

The application architecture is now highly available. However, the Amazon RDS database operates from only one database instance.



Step 7.1: On the AWS Management Console, in the search box, enter and choose RDS to open the Amazon **RDS** console. Then, In the left navigation pane, choose **Databases**.

Step 7.2: Choose the link for the name of the **inventory-db** instance and select **Modify**

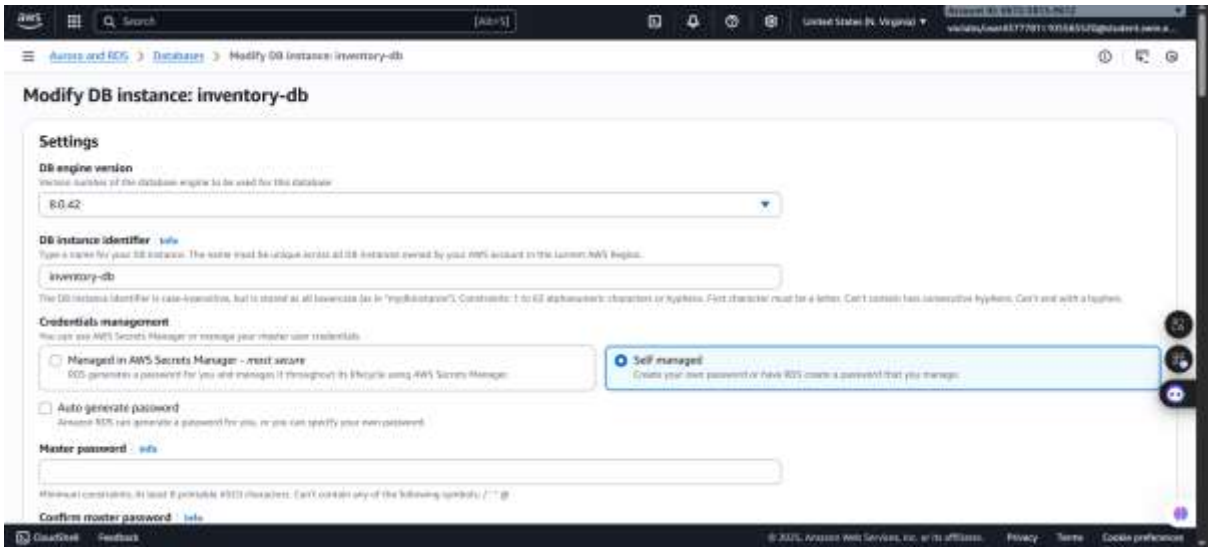


Figure 52: Modify configuration page

Step 7.3: In the **Availability & durability** section, for **Multi-AZ deployment**, choose **Create a standby instance**.

Figure 53: Availability and durability configuration

Step 7.4: In the **Instance configuration** section, for **DB instance class**, choose **db.t3.small.**

Figure 54: Instance configuration

Step 7.5: In the **Storage** section, for **Allocated storage**, enter *20.*

Figure 55: Stroage configuration

Step 7.6: At the bottom of the page, choose **Continue.**

Step 7.7: For **Schedule modifications**, choose **Apply immediately**.

Figure 56: Schedule modifications configuration

Step 7.8: Choose **Modify DB instance.**
The Status of the database is Modifying while it applies the changes. You do not need to wait for it to complete.

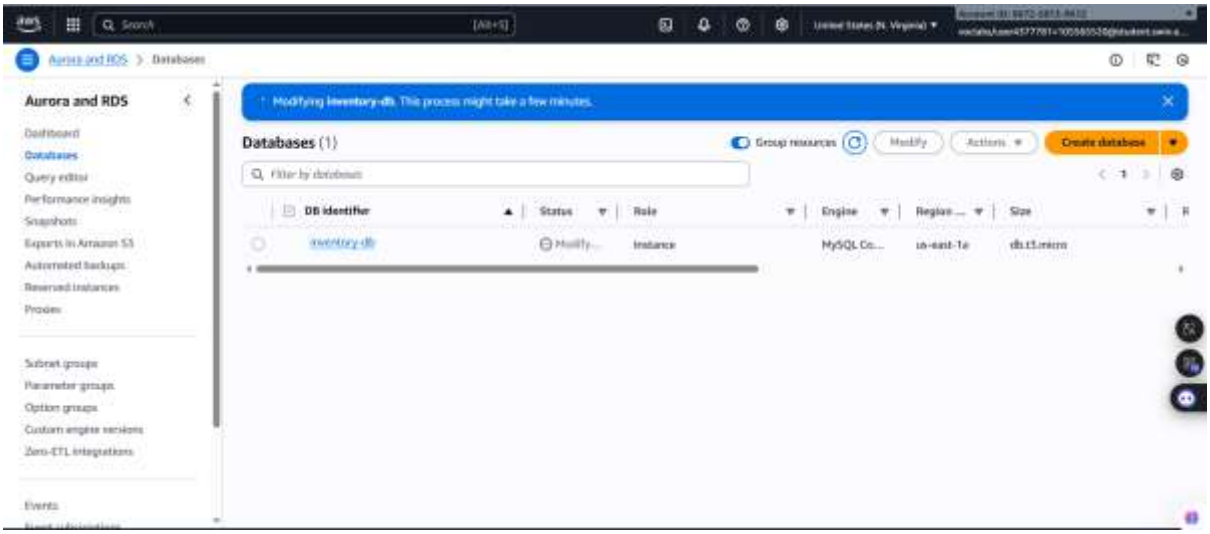
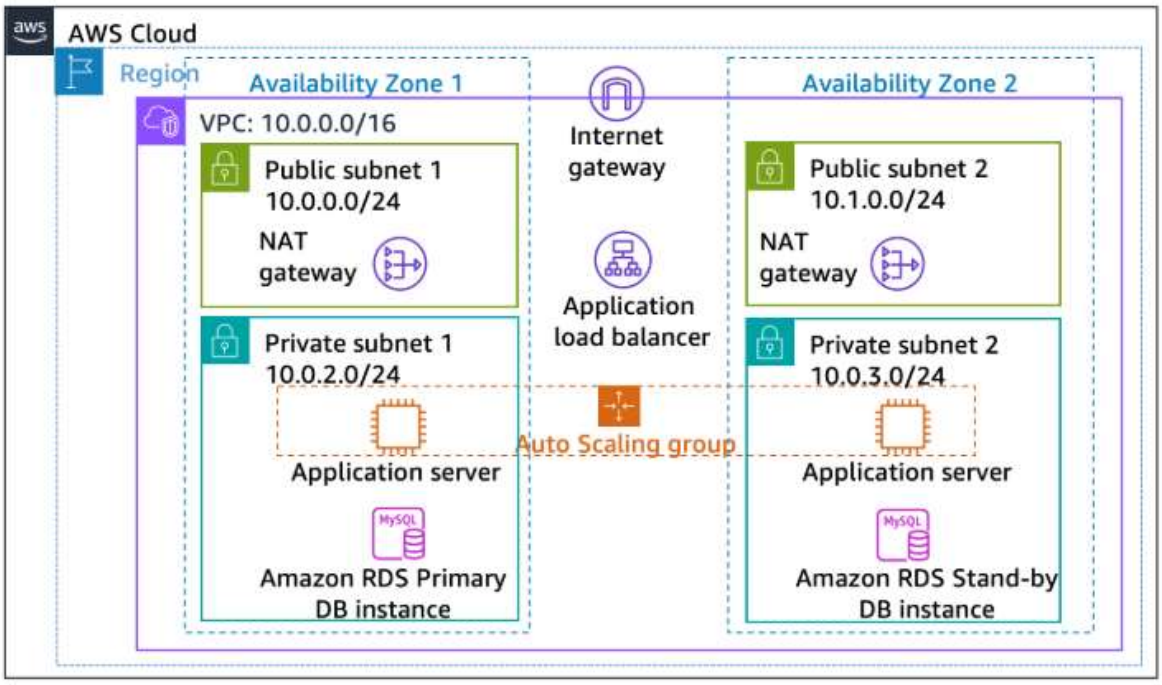


Figure 57: Modifying

I. Task 8: Configuring a highly available NAT gateway (Optional)



Step 8.1: On the AWS Management Console, in the search box, enter and choose **VPC** to open the Amazon VPC console. Next, in the left navigation pane, choose **NAT gateways**.

Step 8.2: Choose **Create NAT gateway.**

Step 8.3: On the Create NAT gateway page, configure the following options*:*

- For **Name -optional**, enter NatGateway2NatGateway2.
- For **Subnet,** choose **Public Subnet 2.**
- Choose **Allocate Elastic IP**.

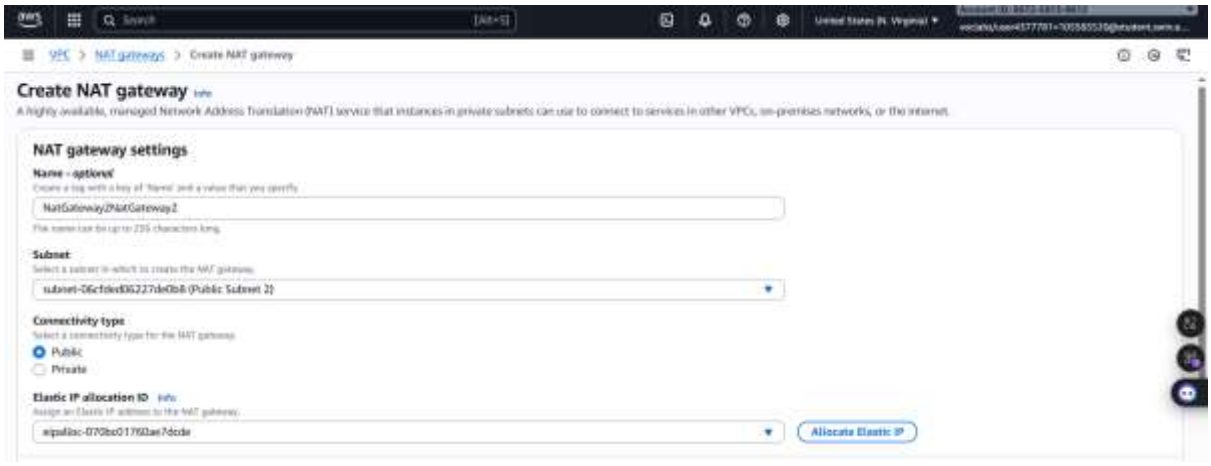


Figure 58: NAT gateway settings configuration

Step 8.4: Choose **Create NAT gateway**.

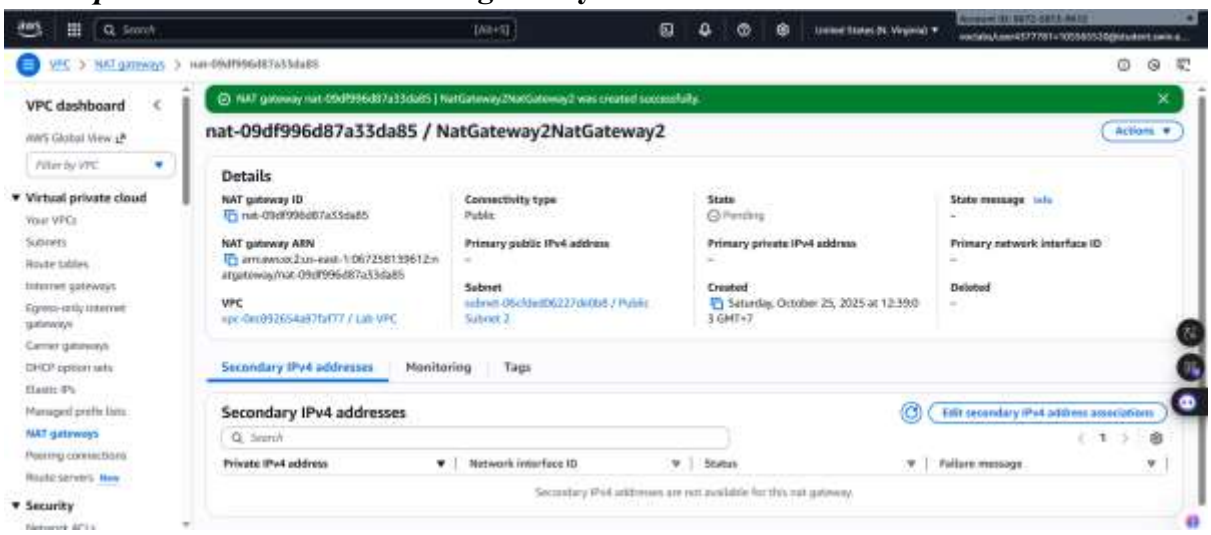


Figure 59: NAT gateway create successfully

Step 8.5: In the left navigation pane, choose **Route tables** and choose **Create route table**.

Step 8.6: On the Create route table page, configure the following options:

- **Name:** Private Route Table 2
- **VPC: Lab VPC**

After configure above settings, Choose **Create route table**

Figure 60: Route table configuration

Step 8.6: Choose the **Routes** tab, and observe the settings. Next, Choose **Edit routes**.

Step 8.7: On the **Edit routes** page, configure the following options:

- Choose **Add route**.
- For **Destination**, enter 0.0.0.0/0.
- For **Target**, choose **NAT Gateway**, and then choose **NatGateway2**.

Choose **Save changes**.

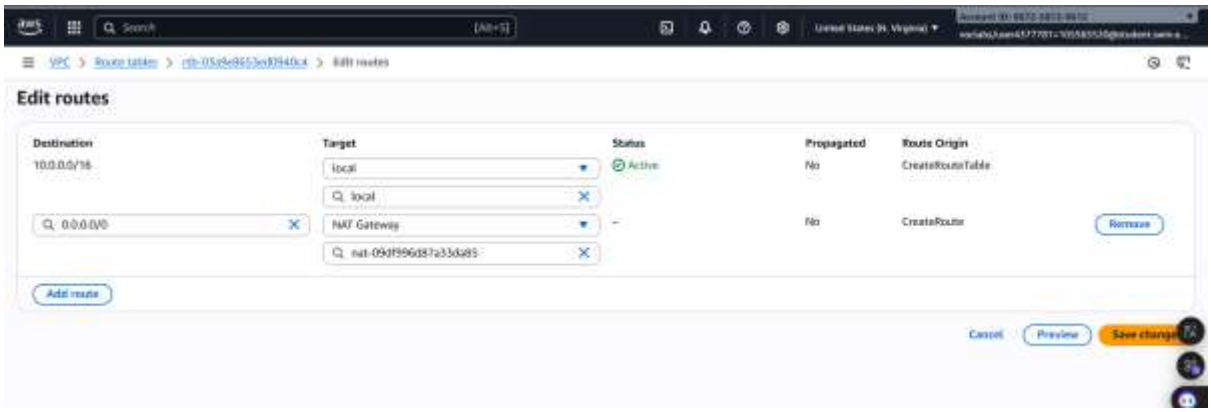


Figure 61: Edit route configuration

Step 8.8: Choose the **Subnet associations** tab, select **Edit subnet associations, Select Private Subnet 2 .**

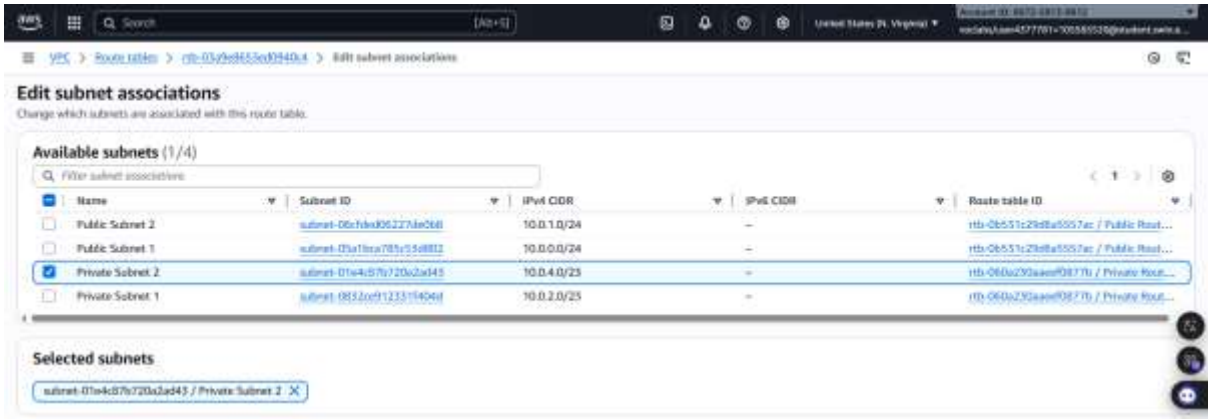
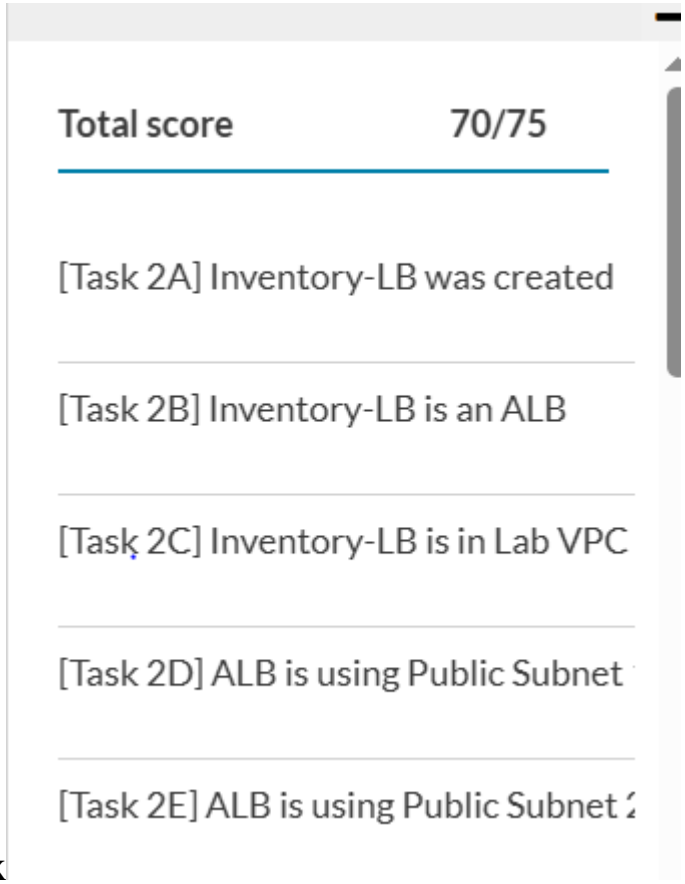


Figure 62: Edit subnet associations configuration



J. Submitting your work

Figure 63: Result

K. Conclusion

In conclusion, **I successfully completed the Guided Lab on creating a highly available environment**, transforming an application running on a single Amazon EC2 instance into a resilient and scalable system. The primary goal of achieving **high availability (HA)** was met by deploying the application across multiple **Availability Zones (AZs)** within the provided Virtual Private Cloud (VPC).

I began by inspecting the existing VPC configuration. Subsequently, **I configured an Application Load Balancer (ALB)** and created an **Auto Scaling group**. By integrating these services, I was able to distribute incoming traffic efficiently and ensure that the application could automatically scale to meet demand and replace failed instances. Ultimately, I was able to **test the application for high availability**, confirming its ability to remain operational and accessible even after deliberately terminating one of the running EC2 instances.

This lab provided me with valuable, hands-on experience in implementing fundamental cloud architecture principles. It reinforced my understanding of how AWS services like the ALB and Auto Scaling are critical components for building **fault-tolerant** and **elastic** applications, which is essential for any critical business system in the cloud.