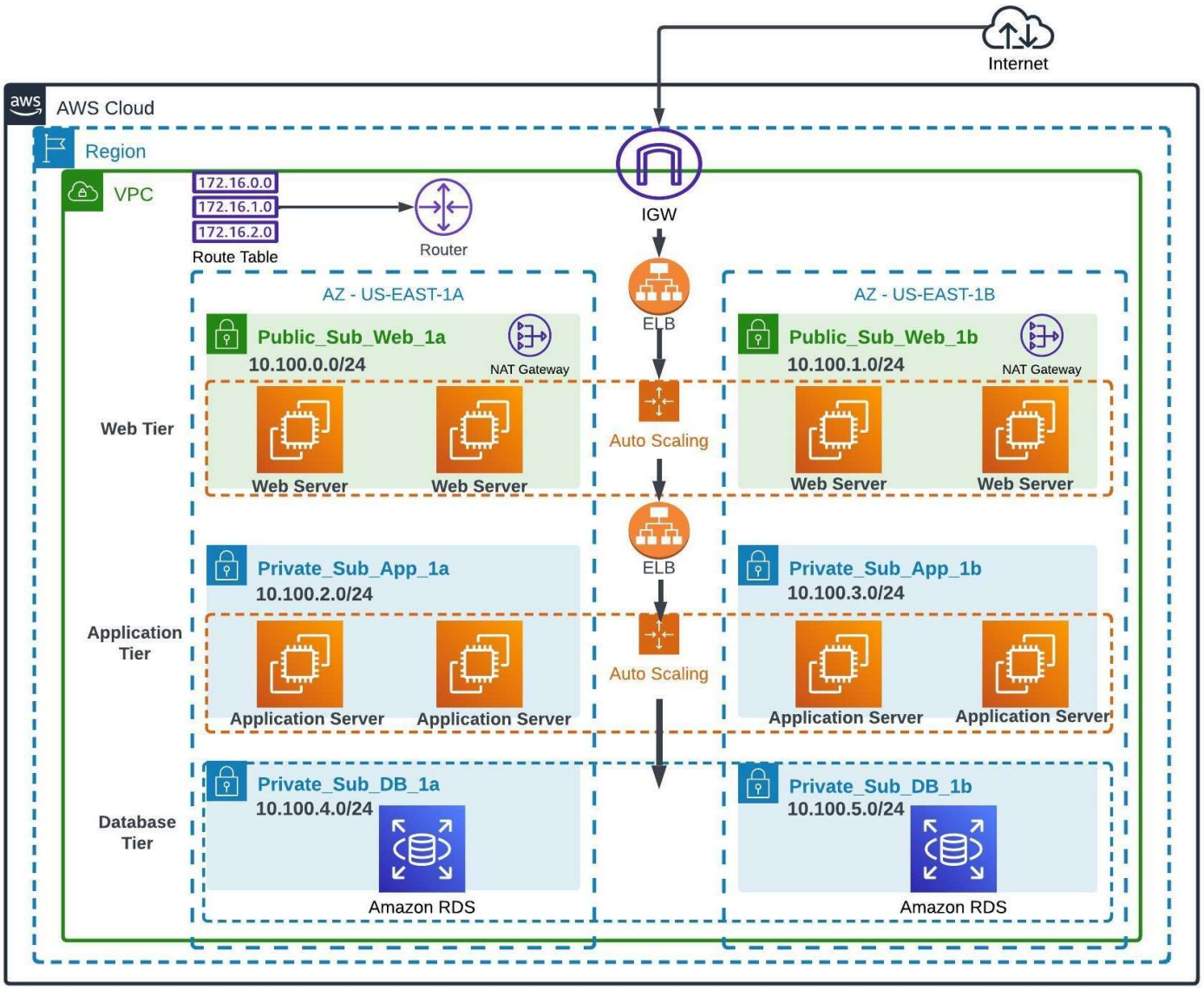
CREATION OF A HIGHLY AVAILABLE 3-TIER ARCHITECTURE

AWS 3-Tier Architecture is made up of 3 separate tiers (hence the name) and they are: The Web Tier, The Application Tier, and Database Tier.



HIGHLY AVAILABLE AWS 3-TIER ARCHITECTURE

AWS 3-TIER ARCHITECTURE:

A 3-Tier architecture is made up of 3 separate tiers or layers, which are the Presentation or Web layer, The Application layer, and Database layer.

The main benefit to this architecture is that it scales horizontally as needed by spinning up more EC2 instances to meet demand whenever the need to scale arises.

Since the architecture consist of multiple EC2 Instances running in an Auto-Scaling Group (ASG) behind an Application Load Balancer (ALB), and across multiple availability zones (AZs), the risk of a single point of failure that might result to undesired down-time is minimized, because in the event that an AZ goes down, the system will failover to the another Availability Zone (AZ) and continue running without disruption.

WEB TIER:

The web tier also known as the presentation tier is the front end, that displays information in the form of GUI to users, by communicating with the application tier through application program interface (API) calls.

REQUIREMENTS:

1. 2 public subnets

2. Minimum of 2 EC2 instances with an OS of your choice (free tier) in an Auto Scaling Group.

3. EC2 web server security group allowing inbound permission from the internet.

4. Boot strap static web page or create a custom AMI that already includes the static web page.

5. Create a public route table and associate the 2 public subnets.

APPLICATION TIER:

The application tier is the man in the middle or connecting bridge between the presentation/web tier and the database tier. The responsibility of the application tier is to collect the data from the presentation tier and process it against the information in the data tier by using API calls.

Since the presentation layer and data layer cannot communicate directly, all communication between the presentation tier and database tier is bridged by the application layer.

REQUIREMENTS:

1. Two private subnets

2. Minimum of 2 EC2 instances with an OS of your choice (free tier) in an Auto Scaling Group.

3. EC2 application server security group allowing inbound permission from the web server security group.

4. Associate with private route table.

Note: This is not a true application tier as we don't have any provided code to run on the EC2 instances.

DATABASE TIER:

The data tier, also known as the backend or storage tier, is where data resources are stored and managed. Some examples of fully managed Relational Database Service (RDS), which is a managed database service provided by Amazon are (1) Amazon RDS for MySQL, (2) Amazon RDS for PostgreSQL (3) Amazon RDS for Oracle (4) Amazon RDS for SQL Server and (5) Amazon Aurora, which is a MySQL and PostgreSQL-compatible relational database service that offers high performance, scalability, and availability.

REQUIREMENTS:

1. Use a free Tier MySQL RDS Database.

2. The Database Security Group should allow inbound traffic for MySQL from the Application Server Security Group.

3. 2 private subnets.

4. Associate with private route table

SCENERIO:

Your manager assigned you the task of designing a highly available 3 Tier architecture web application for a new client.

IMPORTANT:

When building a 3-Tier architecture, It is crucial to always implement the AWS pillars of The Well Architected Framework which include:

1.Availability

2.Security

3.Performance

4.Reliability

5.Cost Optimization

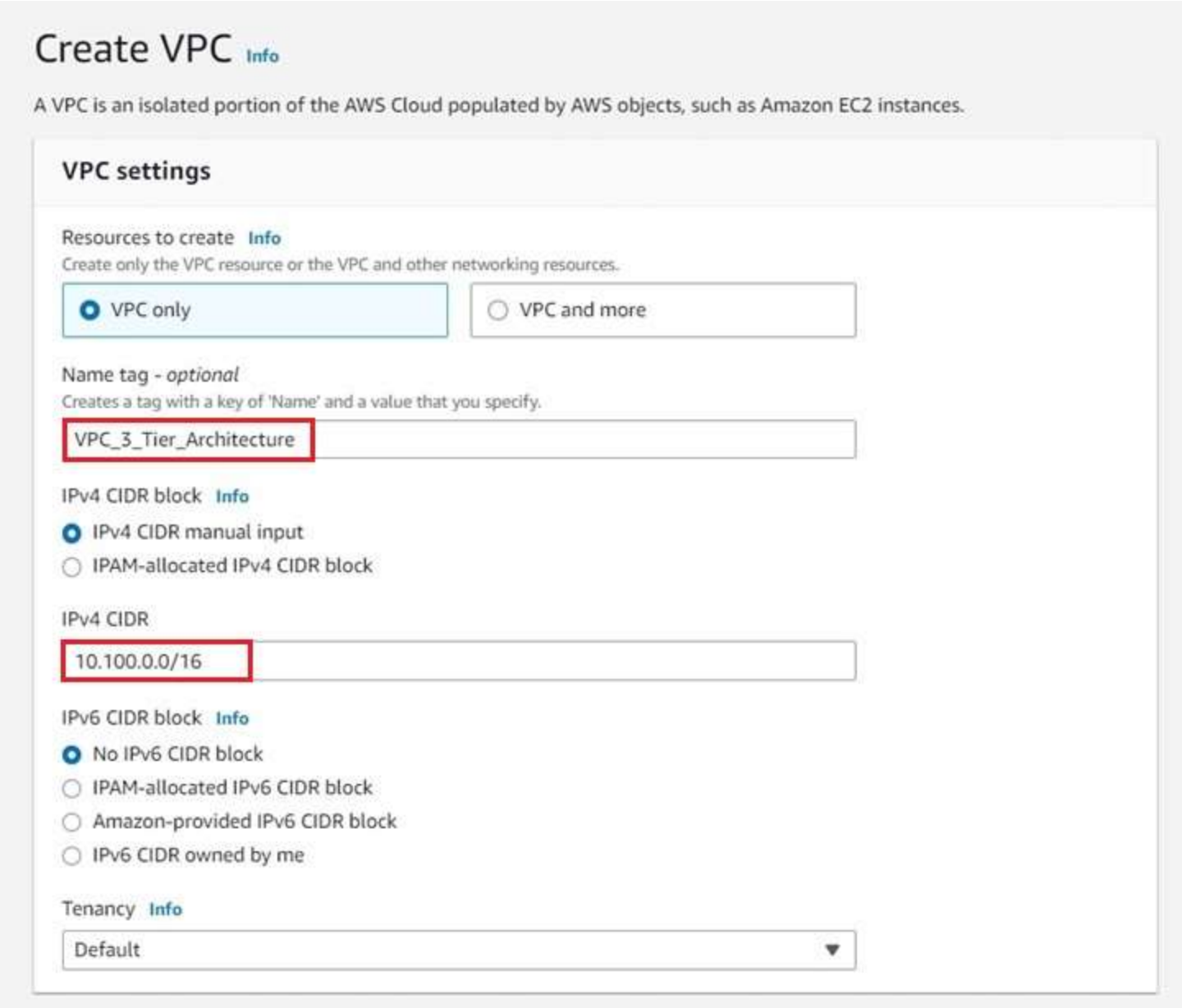
LADIES AND GENTLEMENT, IT'S SHOWTIME, SO LET'S DIVE IN AND GET OUR HANDS DIRTY!

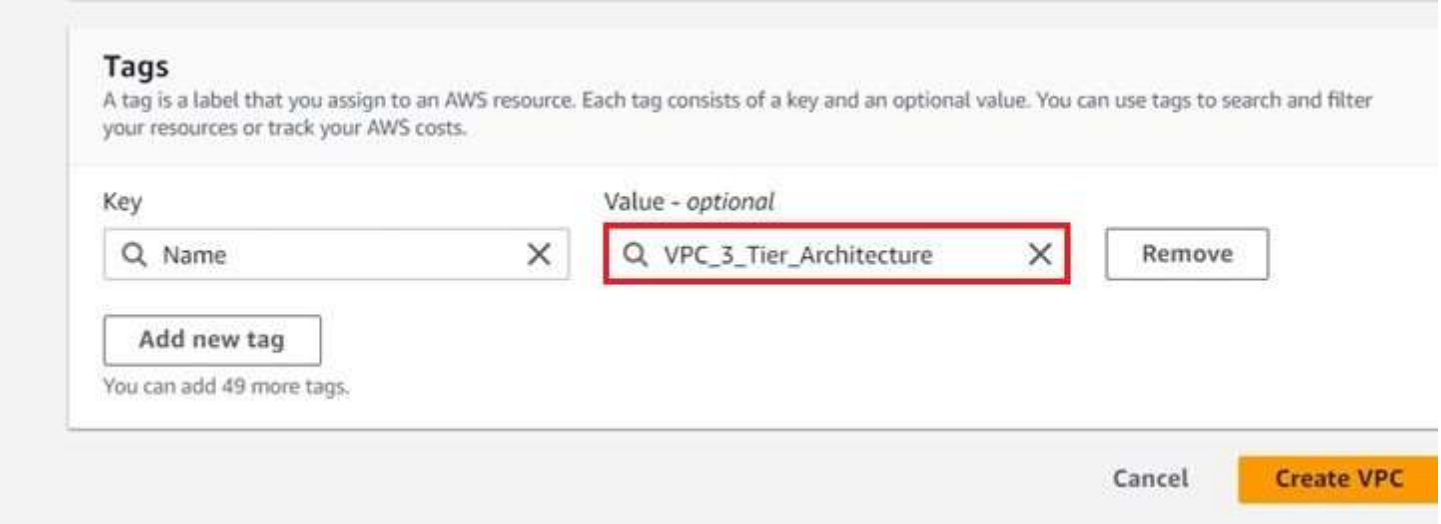
WEB TIER

STEP 1: WE HAVE TO CREATE A VIRTUAL PRIVATE CLOUD (VPC)

First, let's create a VPC.

We also must specify the range of IPv4 CIDR block address for the VPC, and my specified CIDR range will be **10/100.0.0/16**.





My VPC_3-Tier_Architecture was successfully created.

STEP 2: NEXT WE NEED TO CREATE 6 SUBNETS (2 PUBLIC and 4 PRIVATE)

Next, we need to create 6 Subnets, we can do this by specifying a subnet name, the availability zone and the IPv4 CIDR block for each subnet.

The CIDR blocks assigned to the subnets must be derived from my main VPC CIDR block **10.100.0.0/16.**

VPC > Subnets > Create subnet

Create subnet Info

VPC

VPC ID
Create subnets in this VPC.

vpc-08328352666abbbb8 (VPC_3_Tier_Architecture)

Associated VPC CIDRs

IPv4 CIDRs

10.100.0.0/16

Subnet settings

Specify the CIDR blocks and Availability Zone for the subnet.

Subnet 1 of 6

Subnet name
Create a tag with a key of 'Name' and a value that you specify.

Public_Sub_Web_1a

The name can be up to 256 characters long.

Availability Zone Info
Choose the zone in which your subnet will reside, or let Amazon choose one for you.

US East (N. Virginia) / us-east-1a

IPv4 CIDR block Info

10.100.0.0/24

▼ Tags - *optional*

Key	Value - *optional*	
Name	Public_Sub_Web_1a	Remove

Add new tag

You can add 49 more tags.

Remove

Subnet 2 of 6

Subnet name

Create a tag with a key of 'Name' and a value that you specify.

Public_Sub_Web_1b

The name can be up to 256 characters long.

Availability Zone **Info**

Choose the zone in which your subnet will reside, or let Amazon choose one for you.

US East (N. Virginia) / us-east-1b

IPv4 CIDR block **Info**

10.100.1.0/24

▼ Tags - *optional*

Key	Value - *optional*	
Name	Public_Sub_Web_1b	Remove

Add new tag

You can add 49 more tags.

Remove

Subnet 3 of 6

Subnet name

Create a tag with a key of 'Name' and a value that you specify.

Private_Sub_App_1a

The name can be up to 256 characters long.

Availability Zone **Info**

Choose the zone in which your subnet will reside, or let Amazon choose one for you.

US East (N. Virginia) / us-east-1a

IPv4 CIDR block **Info**

10.100.2.0/24

▼ Tags - *optional*

Key	Value - *optional*	
Name	Private_Sub_App_1a	Remove

Add new tag

You can add 49 more tags.

Remove

Subnet 4 of 6

Subnet name
Create a tag with a key of 'Name' and a value that you specify.
Private_Sub_App_1b
The name can be up to 256 characters long.

Availability Zone Info
Choose the zone in which your subnet will reside, or let Amazon choose one for you.
US East (N. Virginia) / us-east-1b

IPv4 CIDR block Info
10.100.3.0/24

▼ Tags - *optional*

Key	Value - *optional*	
Name	Private_Sub_App_1b	Remove

Add new tag
You can add 49 more tags.

Remove

Subnet 5 of 6

Subnet name
Create a tag with a key of 'Name' and a value that you specify.
Private_Sub_DB_1a
The name can be up to 256 characters long.

Availability Zone Info
Choose the zone in which your subnet will reside, or let Amazon choose one for you.
US East (N. Virginia) / us-east-1a

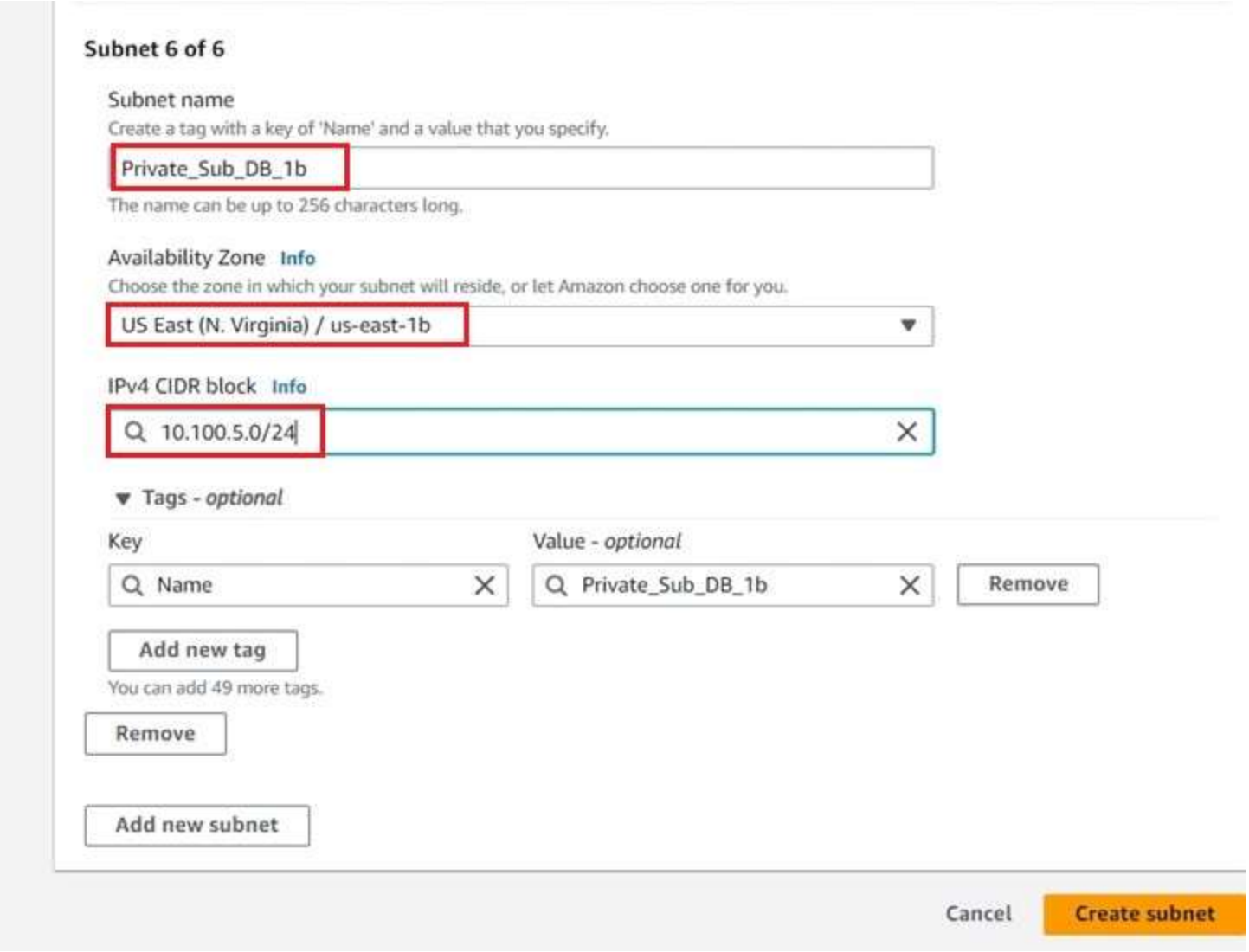
IPv4 CIDR block Info
10.100.4.0/24

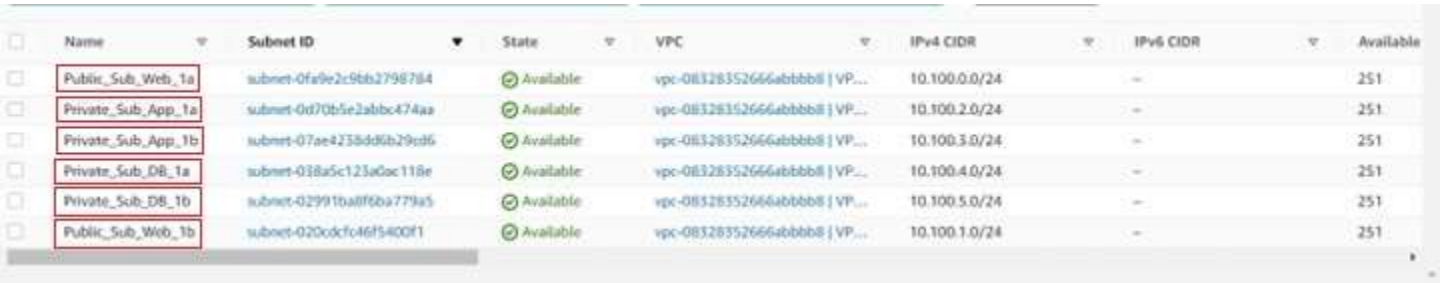
▼ Tags - *optional*

Key	Value - *optional*	
Name	Private_Sub_DB_1a	Remove

Add new tag
You can add 49 more tags.

Remove



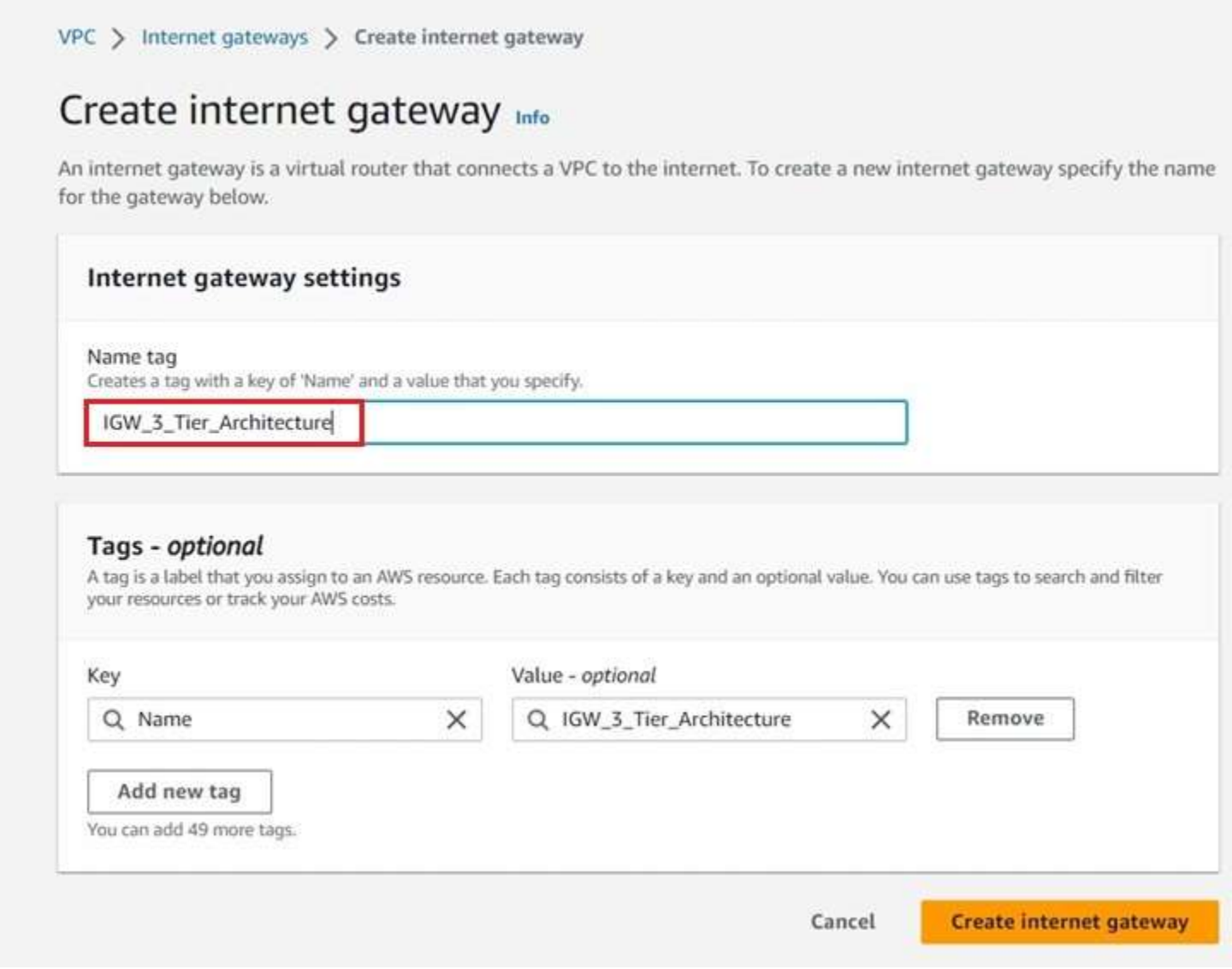


All 6 subnets have been successfully created as we can see from the above screenshots.

It is important to note that the public subnets (**Public_Sub_Web_1a**) and (**Public_Sub_Web_1b**) will be used in the **Web Tier**.

STEP 3: NOW, WE HAVE TO CREATE AN INTERNET GATEWAY (IGW)

Let's create an Internet Gateway (**IGW**), after that we must select my VPC and then attach the internet gateway to the VPC.



Internet gateway **"IGW_3_Tier_Architecture"** has been successfully created and attached to our VPC.

STEP 4: NEXT WE HAVE TO CONFIGURE TWO (2) NAT GATEWAYS

Let’s create two (2) NAT Gateways for redundancy.

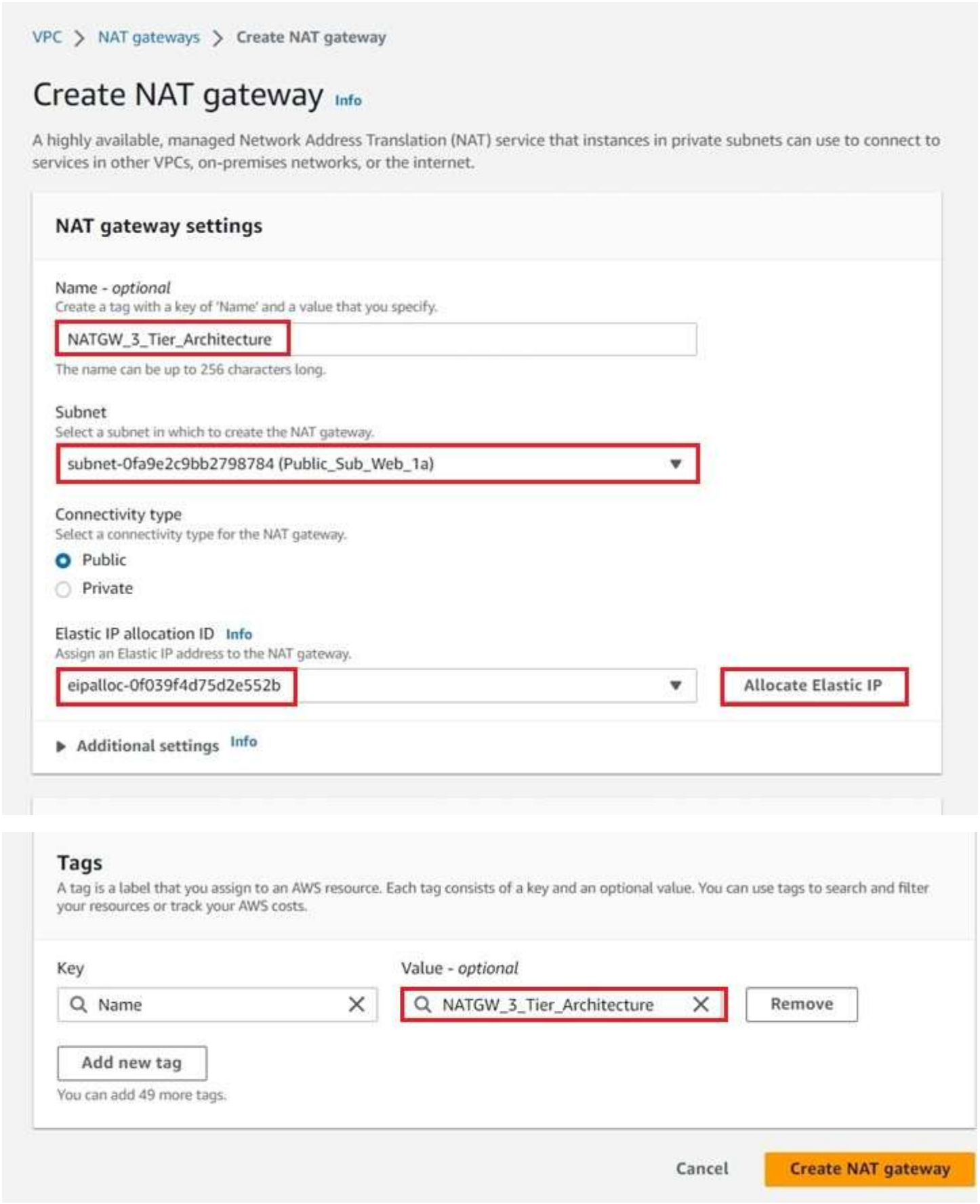
The main function of the NAT Gateway is to allow our EC2 instances in the private subnets to have access to the internet for software updates, package installation, downloads etc.

Navigate to VPC, scroll down to select NAT gateways, and click on create NAT gateway.

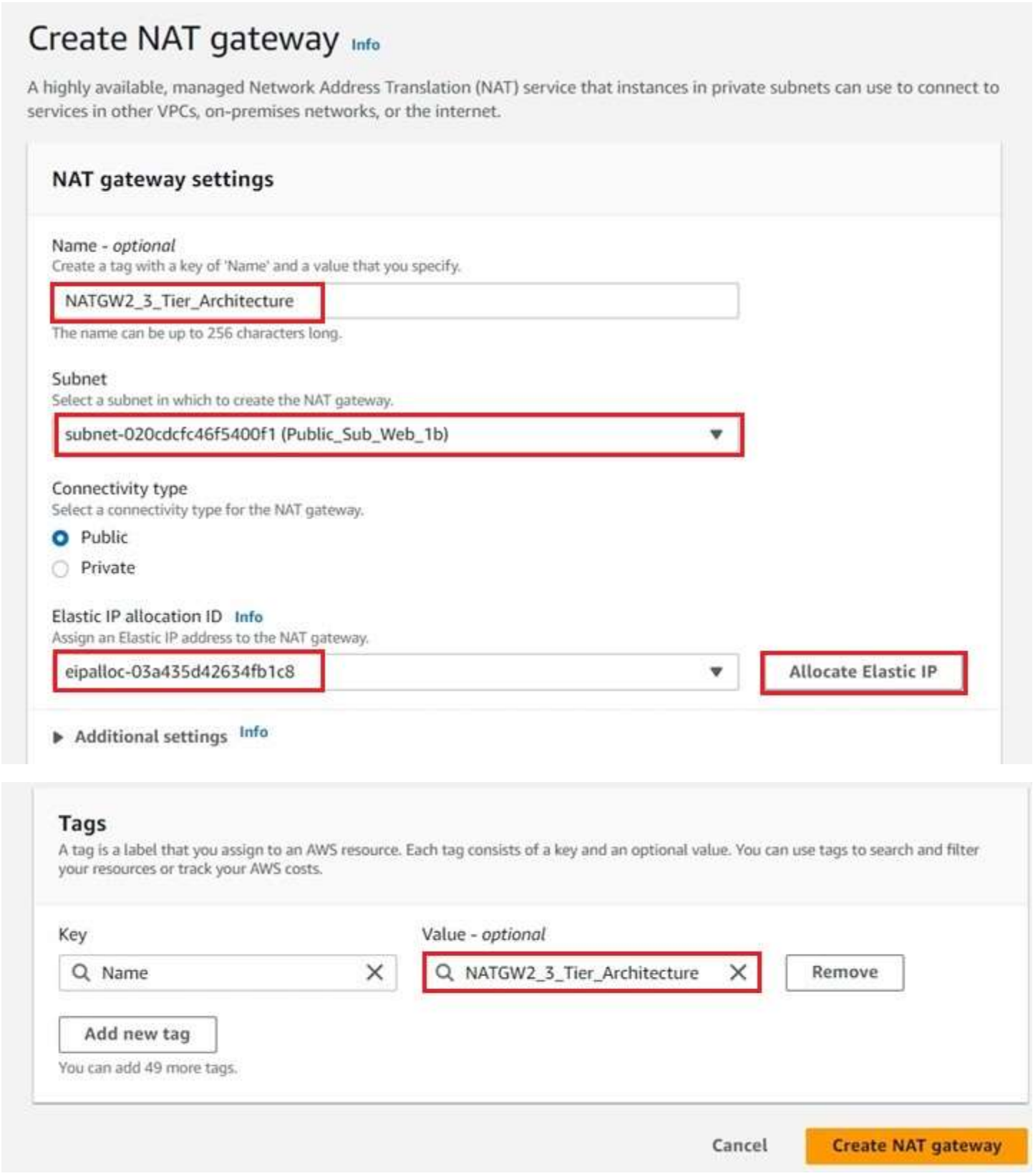
Let’s specify the public subnets where we want the NAT gateways to reside when created. Let’s select our first public subnet **"Public_Sub_Web_1a"** for this configuration, and then select connectivity type as public.

We must click on "**Allocate Elastic IP**" to allocate an Elastic IP address to the NAT gateway.

Click on create NAT gateway and repeat same process to create the second NAT gateway in the second public subnet **"Public_Sub_Web_1b".**



(a) The first NAT gateway **(NATGW_3_Tier_Architecture)** is deployed in the public Web subnet **(Public_Sub_Web_1a)** and mapped to route traffic towards the Internet Gateway (IGW)



(b) The second NAT gateway **(NATGW2_3_Tier_Architecture)** is deployed in the public Web subnet **(Public_Sub_Web_1b)** and mapped to route traffic towards the Internet Gateway (IGW)

Elastic IP is needed because it is a non-changing IP address that is assigned to your resources. If an EC2 instance goes down or terminated accidentally, the assigned elastic IP remains the same when the instance is fired back online.

It is important to always detach and release an elastic IP as soon as you are done with it. An elastic IP when in use, does not incur charges, but it does cost extra charge on your account when not in use and not released to AWS.

STEP 5: CREATION OF ROUTE TABLES

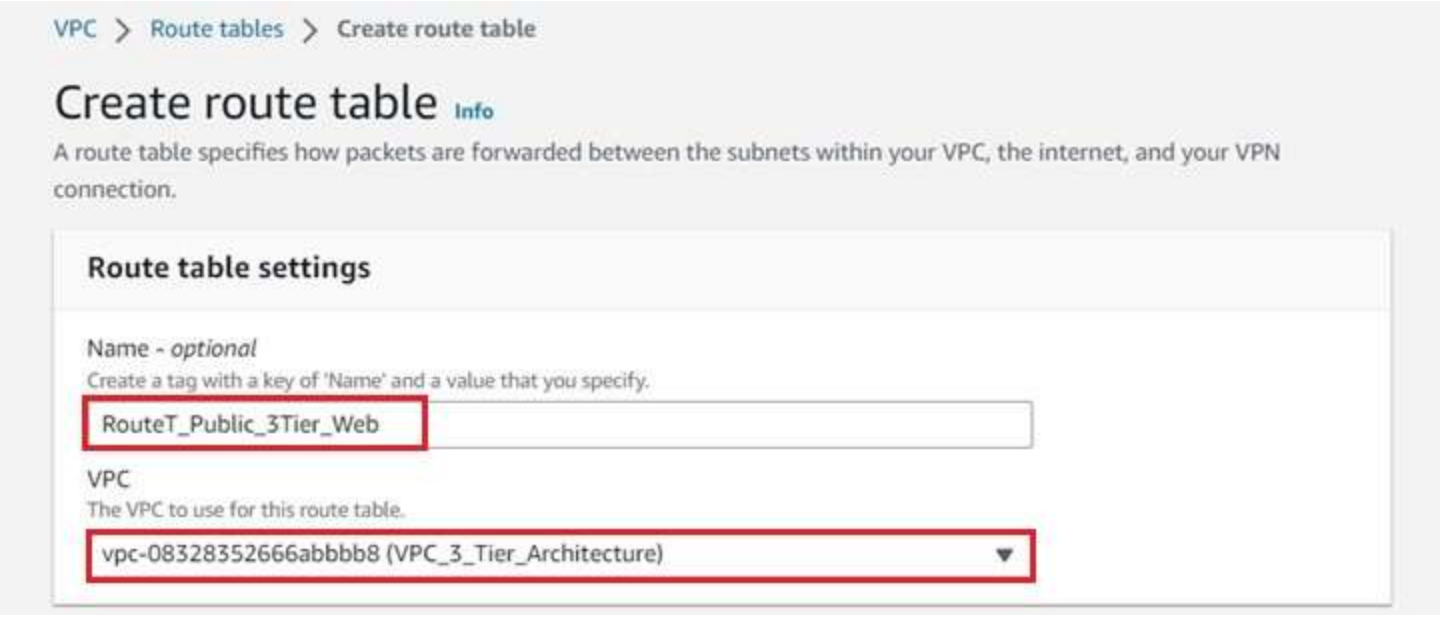
(a) Create Public Route Table **(RouteT_Public_3Tier_Web)**

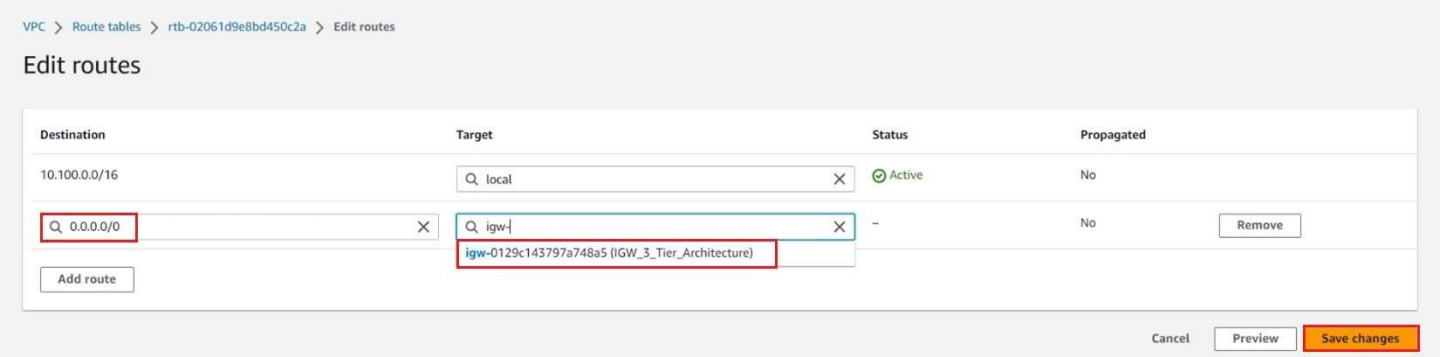
(b) Edit the route table **(RouteT_Public_3Tier_Web)** to route traffic to **Internet Gateway (IGW_3_Tier_Architecture)**

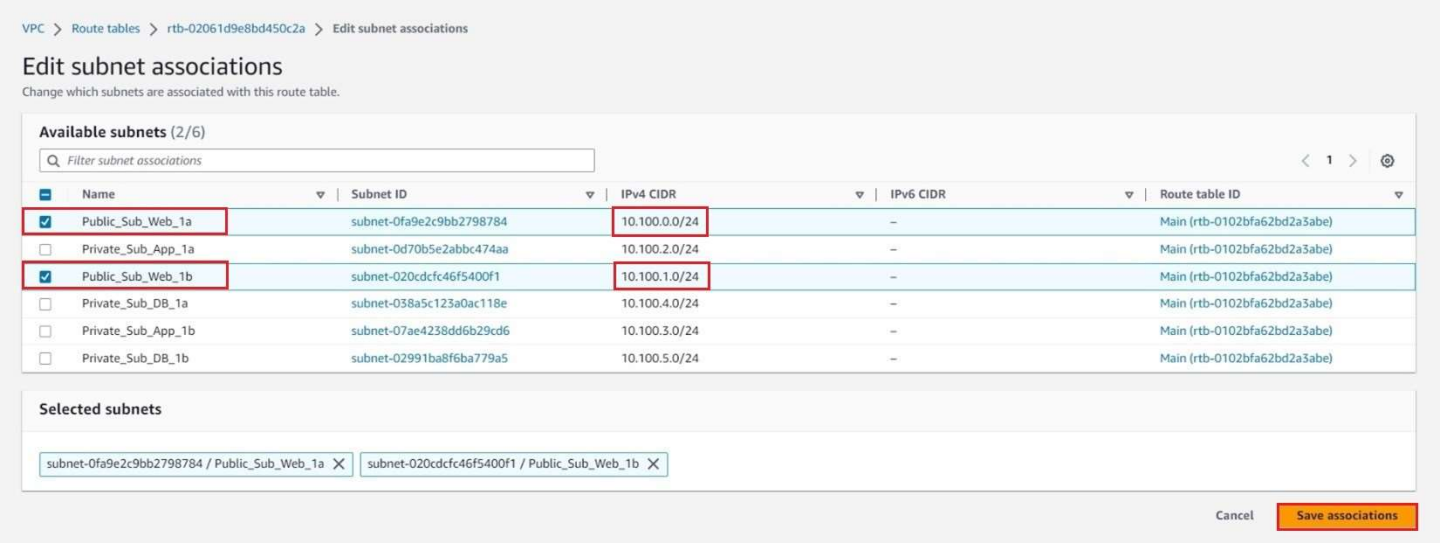
For this configuration, simply click on the Route table **(RouteT_Public_3Tier_Web),** select edit routes, select add route, in the blank field, type in **0.0.0.0/0** and then scroll down to select Internet gateway **(IGW_3_Tier_Architecture)** as target, then click "Save changes".

(c) Edit subnet association

To edit the subnet association of the route table **(RouteT_Public_3Tier_Web)** linked to the two public subnets **(Public_Sub_Web_1a)** and **(Public_Sub_Web_1b),** simply click on "Subnet associations", proceed to click on "Edit subnet associations", select the public subnets **(Public_Sub_Web_1a)** and **(Public_Sub_Web_1b),** then click on "save associations"







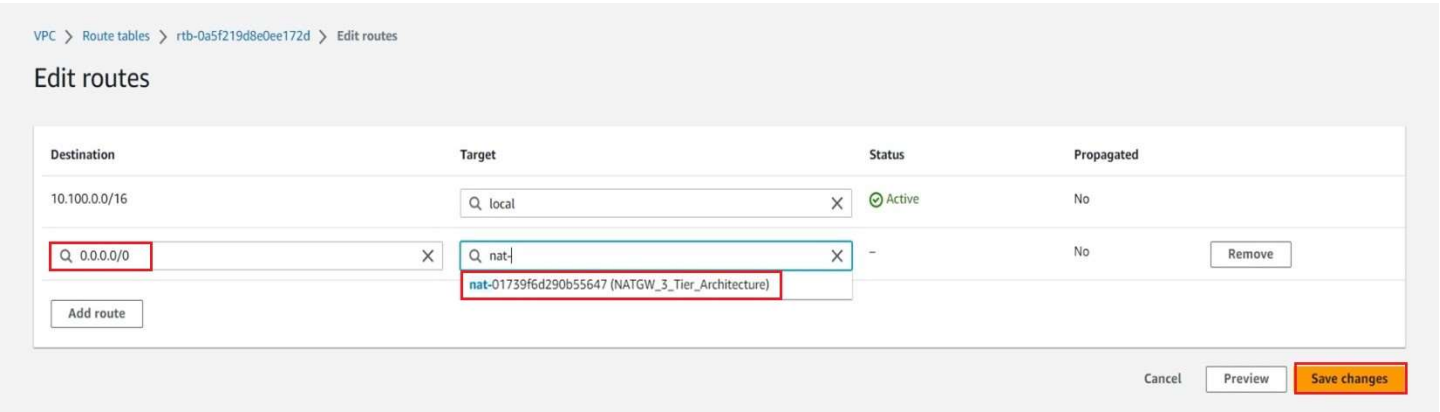
(d) Create the private Route Table **(RouteT_Private_3Tier_App)**

(e) Edit the route table **(RouteT_Private_3Tier_App)** to route traffic to the first NAT gateway **(NATGW_3_Tier_Architecture)** that resides in the public subnet **(Public_Sub_Web_1a).**

For this configuration, simply click on the Route table **(RouteT_Private_3Tier_App),** select edit routes, select add route, in the blank field, type in **0.0.0.0/0** and then scroll down to select NAT **gateway (NATGW_3_Tier_Architecture)** as target, then click "save changes."

(f) Edit subnet association

To edit the subnet association of the route table **(RouteT_Private_3Tier_App)** linked to the 2 private subnets **(Private_Sub_App_1a)** and **(Private_Sub_DB_1a),** simply click on "Subnet associations", proceed to click on "Edit subnet associations", select the private subnets **(Private_Sub_App_1a)** and **(Private_Sub_DB_1a)** then click on "save associations."

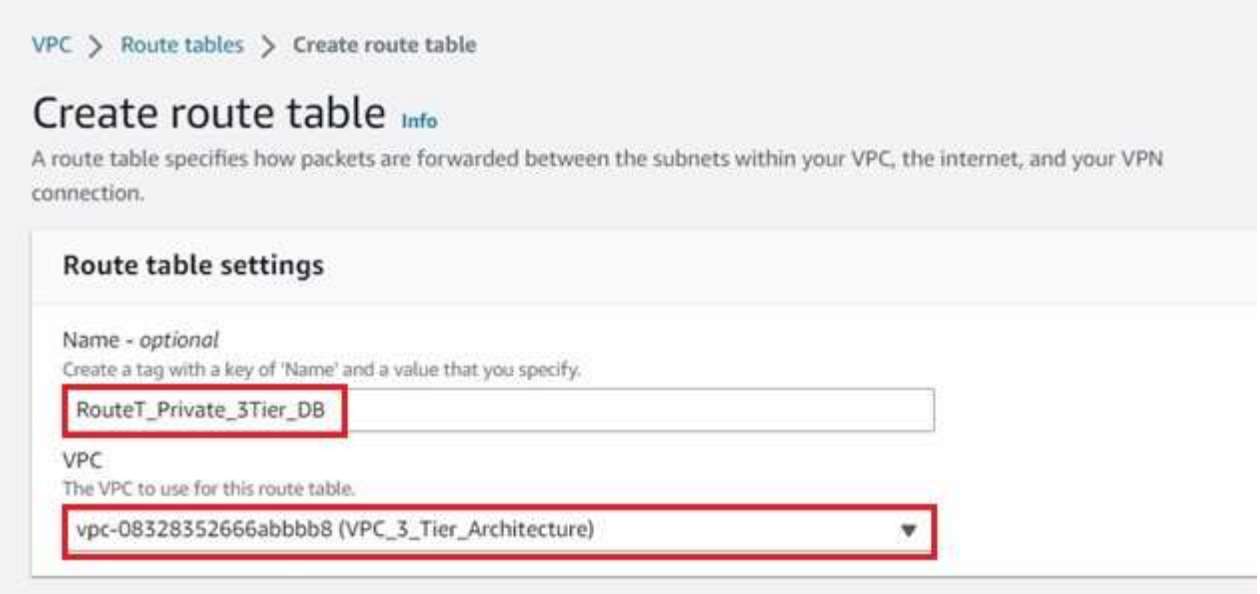


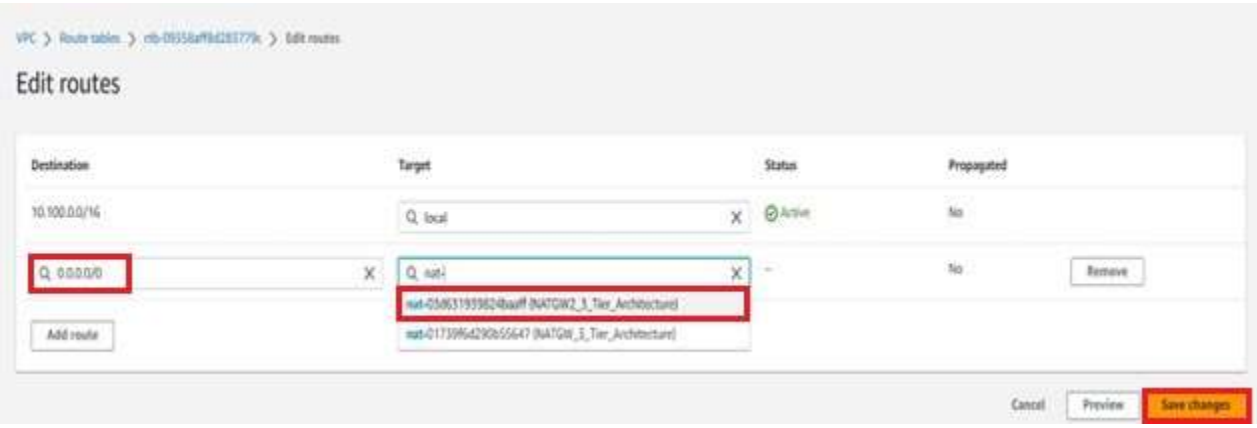
(g) Create the private Route Table **(RouteT_Private_3Tier_DB)**

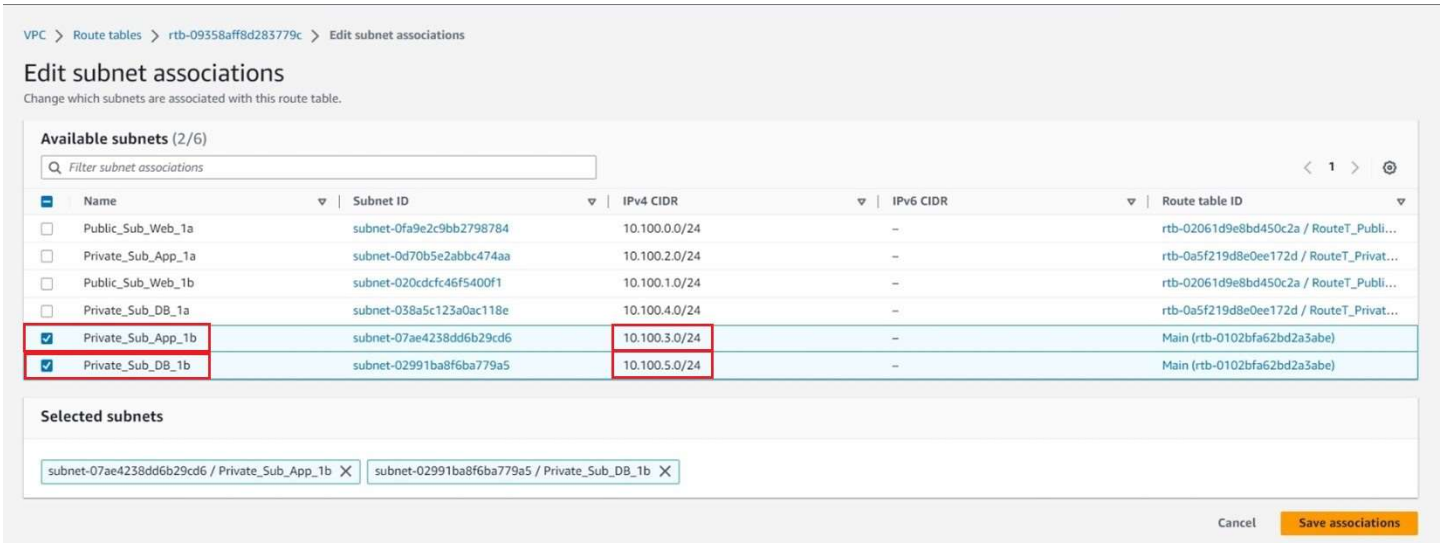
(h) Edit the route table **(RouteT_Private_3Tier_DB)** to route traffic to the second NAT gateway **(NATGW2_3_Tier_Architecture)** that resides in the public subnet **(Public_Sub_Web_1b).** For this configuration, simply click on the Route table **(RouteT_Private_3Tier_DB),** select edit routes, select add route, in the blank field, type in **0.0.0.0/0** and then scroll down to select NAT gateway **(NATGW2_3_Tier_Architecture)** as target, then click "save changes."

(i) Edit subnet association

To edit the subnet association of the route table **(RouteT_Private_3Tier_DB)** linked to the 2 private subnets **(Private_Sub_App_1b)** and **(Private_Sub_DB_1b),** simply click on "Subnet associations", proceed to click on "Edit subnet associations", select the private subnets "**Private_Sub_App_1b** and **Private_Sub_DB_1b**" then click on "save associations."







NOTE: We can associate all four private subnets with one Private Route Table that will point towards any of the NAT gateways, but this all depends on personal choices.

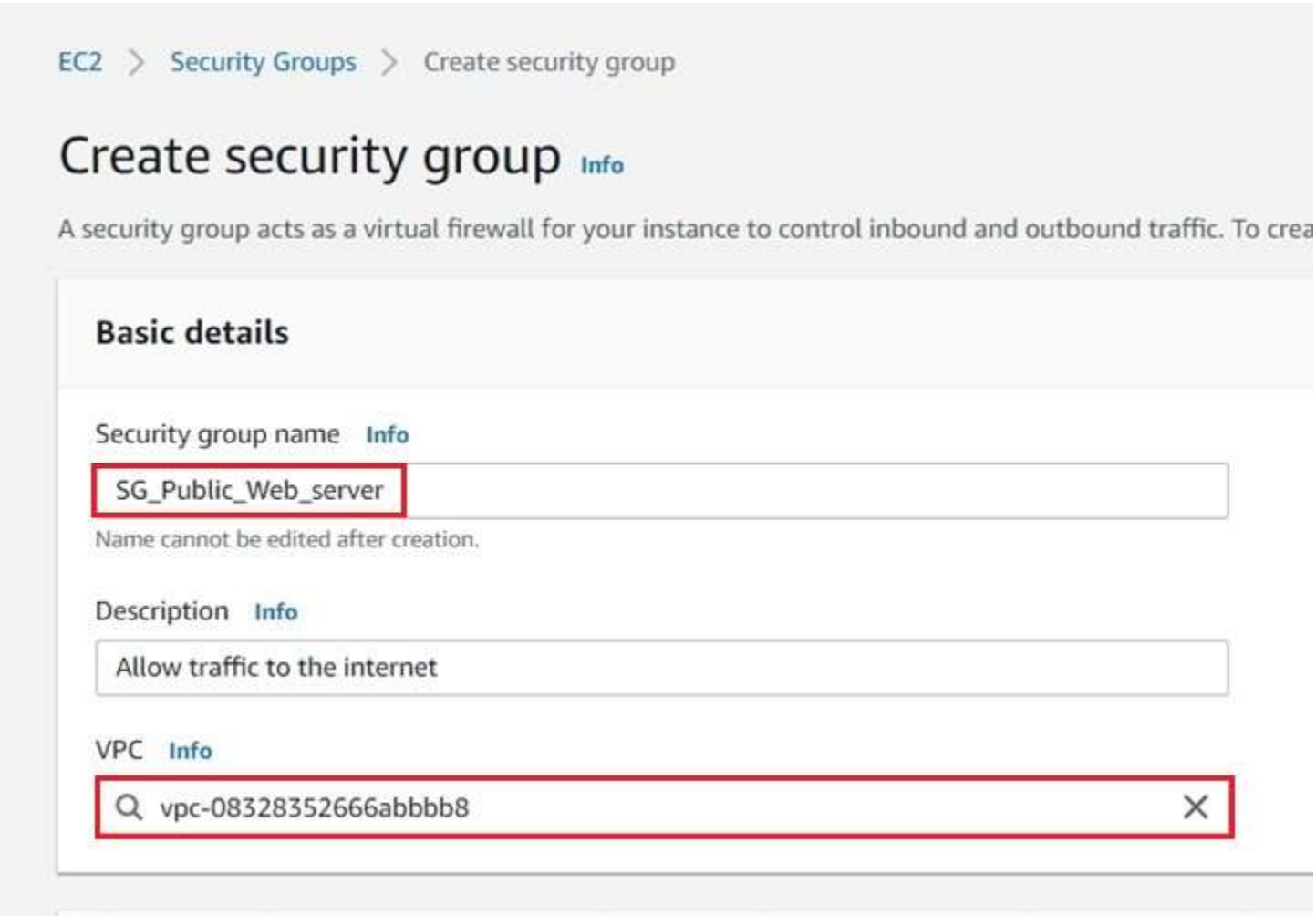
STEP 5: CREATION OF SECURITY GROUPS

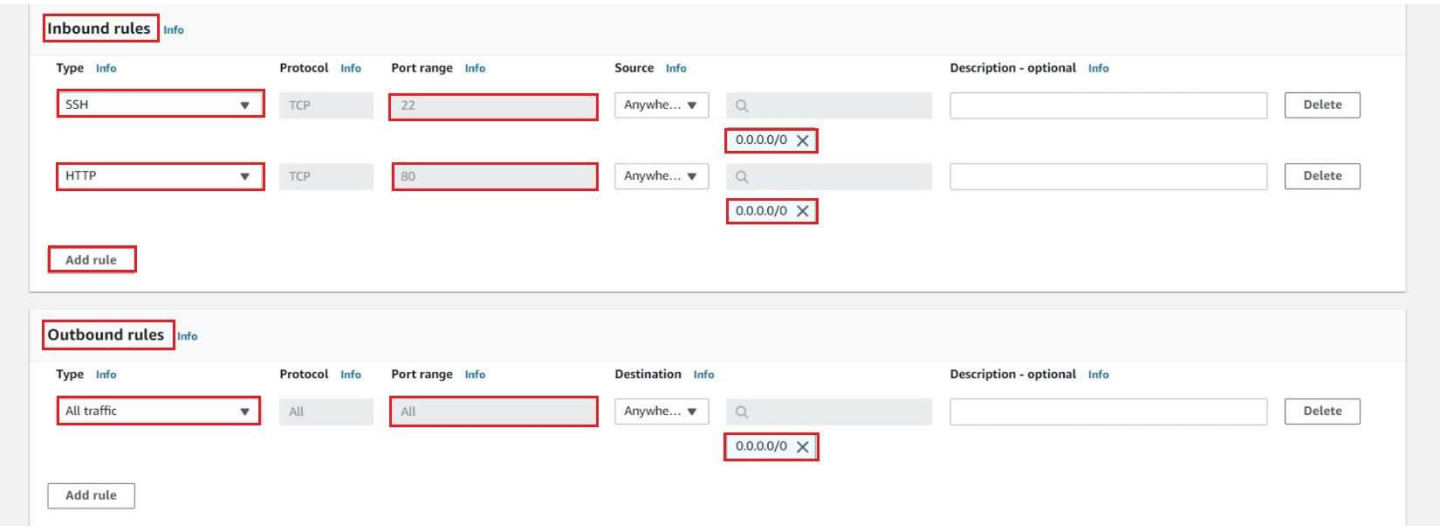
Let's navigate to the EC2 dashboard, scroll down to select Security Groups.

Click on create security group, give it a name, add description, and specify your VPC.

Choose the security group rule that will control our web server inbound and outbound traffic.

(a)Now, we can create our security group.





Security groups are stateful by default, which means it can keep track of the state of network traffic that flows in and out of EC2 instances, and it makes intelligent decisions about allowing or blocking traffic to EC2 instances.

STEP 6: CREATION OF LAUNCH TEMPLATE

A launch template basically streamlines, simplifies, and standardizes the configuration of EC2 instances which are launched by the autoscaling group.

Navigate to the EC2 dashboard, select launch template, and click on create launch template.

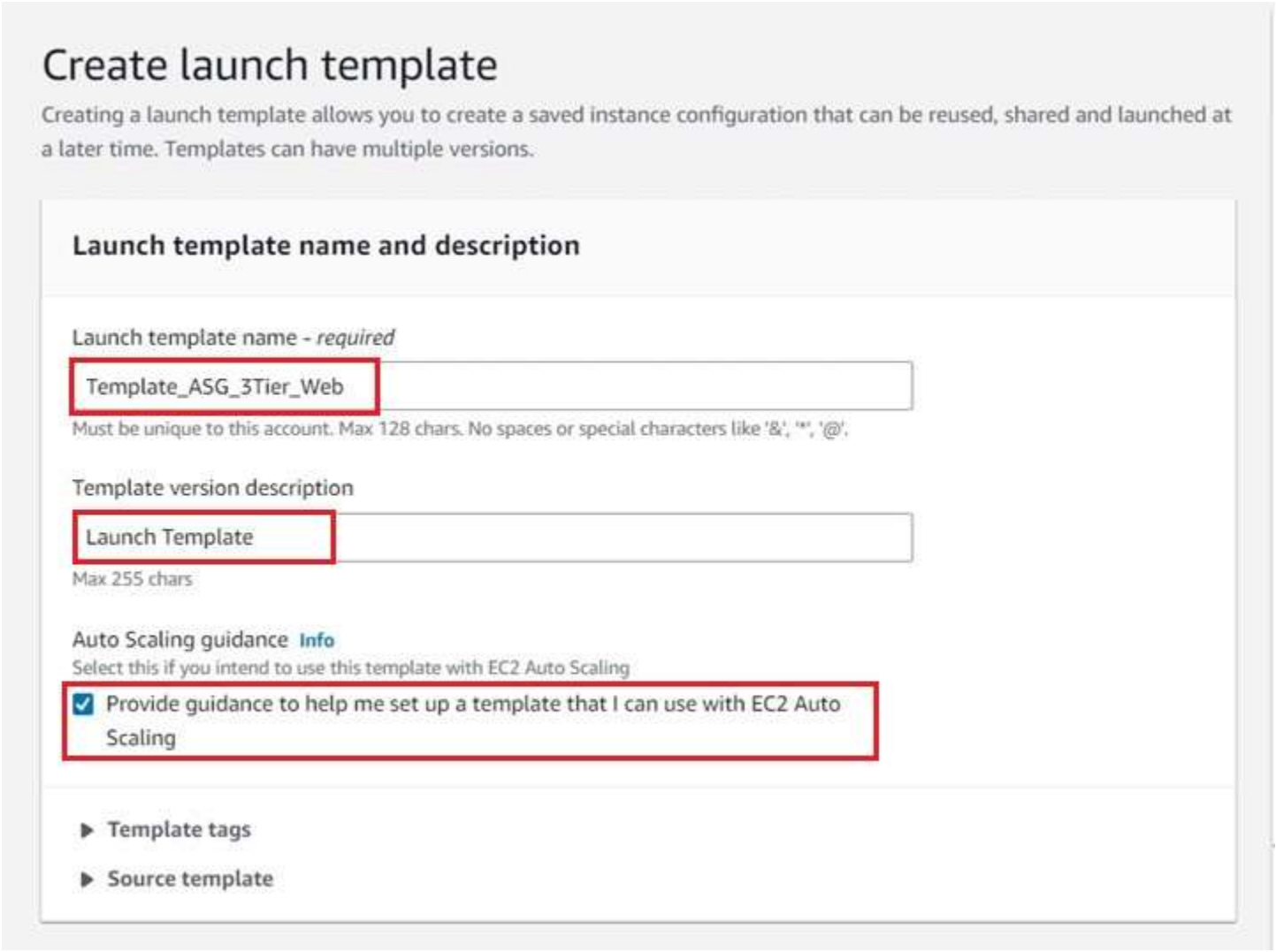
Specify the template name **(Template_3Tier_Web_ASG)**

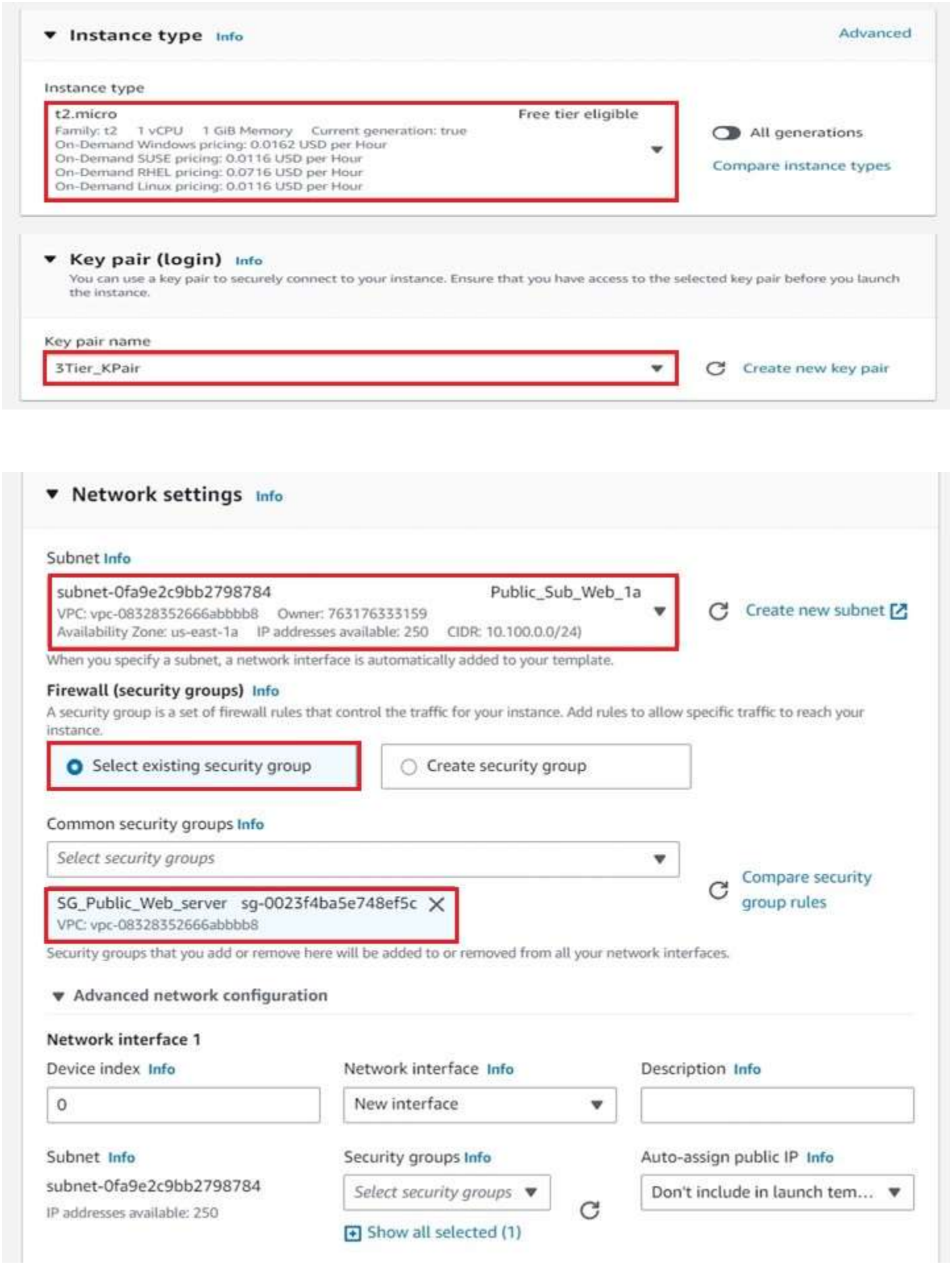
Choose an Amazon machine Image **(Amazon Linux 2, t2. micro),**

Create keypairs or use an existing one **(3Tier_KPair)**

Create Security group **(Template_3Tier_Web_ASG)**

Network configurations: enable auto assign public IP.




Instance type Info
Advanced
Instance type
t2.micro
Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows pricing: 0.0162 USD per Hour
On-Demand SUSE pricing: 0.0116 USD per Hour
On-Demand RHEL pricing: 0.0716 USD per Hour
On-Demand Linux pricing: 0.0116 USD per Hour
All generations
Compare instance types
Key pair (login) Info
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.
Key pair name
3Tier_KPair
Create new key pair
Network settings Info
Subnet Info
subnet-0fa9e2c9bb2798784
Public_Sub_Web_1a
VPC: vpc-08328352666abbbb8 Owner: 763176333159
Availability Zone: us-east-1a IP addresses available: 250 CIDR: 10.100.0.0/24)
Create new subnet
When you specify a subnet, a network interface is automatically added to your template.
Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
Select existing security group
Create security group
Common security groups Info
Select security groups
SG_Public_Web_server sg-0023f4ba5e748ef5c
VPC: vpc-08328352666abbbb8
Compare security group rules
Security groups that you add or remove here will be added to or removed from all your network interfaces.
Advanced network configuration
Network interface 1
Device index Info
0
Network interface Info
New interface
Description Info
Subnet Info
subnet-0fa9e2c9bb2798784
IP addresses available: 250
Security groups Info
Select security groups
Show all selected (1)
Auto-assign public IP Info
Don't include in launch tem...

Add a bootstrap script to user data to launch an Apache webserver.

```
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd
echo "<h1>Hello World, the first part of my 3Tier Web project is resolved!</h1>" >
/var/www/html/index.html
```

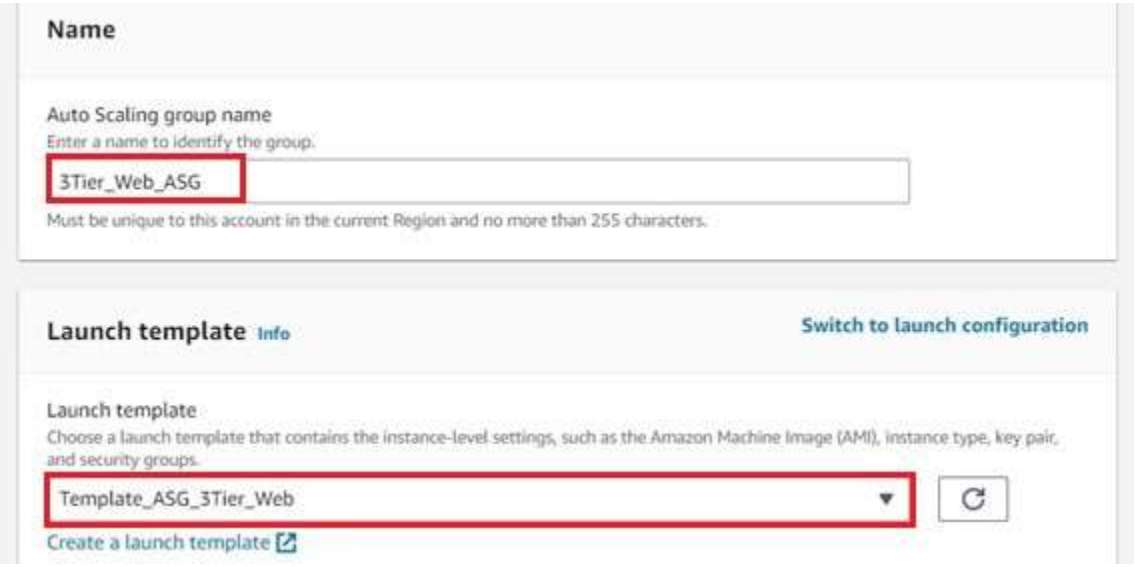
STEP 7: CREATION OF AUTO SCALING GROUP

Now, our next task will be to create an Auto Scaling Group for our Web Tier. This step is straight forward since we already created a launch template for the autoscaling group.

Navigate to the EC2 instance dashboard, scroll down to select Auto Scaling Group.

Add an ALB, that is internet facing and add target group. Traffic will be forwarded to the corresponding target group. Again, make sure the right VPC and subnets are selected.

Click on create, specify a name for the Auto Scaling Group and my ASG for this project is named **(3Tier_Web_ASG).** Choose the previously created launch template **(Template_ASG_3Tier_Web),** scroll down to Network to select VPC, select AZs and desired subnets, click next until end to create auto scaling group.



Choose instance launch options Info

Choose the VPC network environment that your instances are launched into, and customize the instance types and purchase options.

Network Info

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

VPC

Choose the VPC that defines the virtual network for your Auto Scaling group.

vpc-08328352666abbbb8 (VPC_3_Tier_Architec...
10.100.0.0/16

Create a VPC

Availability Zones and subnets

Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.

Select Availability Zones and subnets

us-east-1a | subnet-0fa9e2c9bb2798784 (Public_Sub_Web_1a)
10.100.0.0/24

us-east-1b | subnet-020cdcfc46f5400f1 (Public_Sub_Web_1b)
10.100.1.0/24

Create a subnet

Configure group size and scaling policies - *optional* Info

Set the desired, minimum, and maximum capacity of your Auto Scaling group. You can optionally add a scaling policy to dynamically scale the number of instances in the group.

Group size - *optional* Info

Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum capacity limits. Your desired capacity must be within the limit range.

Desired capacity

2

Minimum capacity

2

Maximum capacity

3

I have configured the auto scaling to have a minimum of 2, a desired capacity of 2 and a maximum capacity of 3 EC2 instances for the auto scaling group.

STEP 8: CREATE TARGET GROUP

Go to the EC2 service.

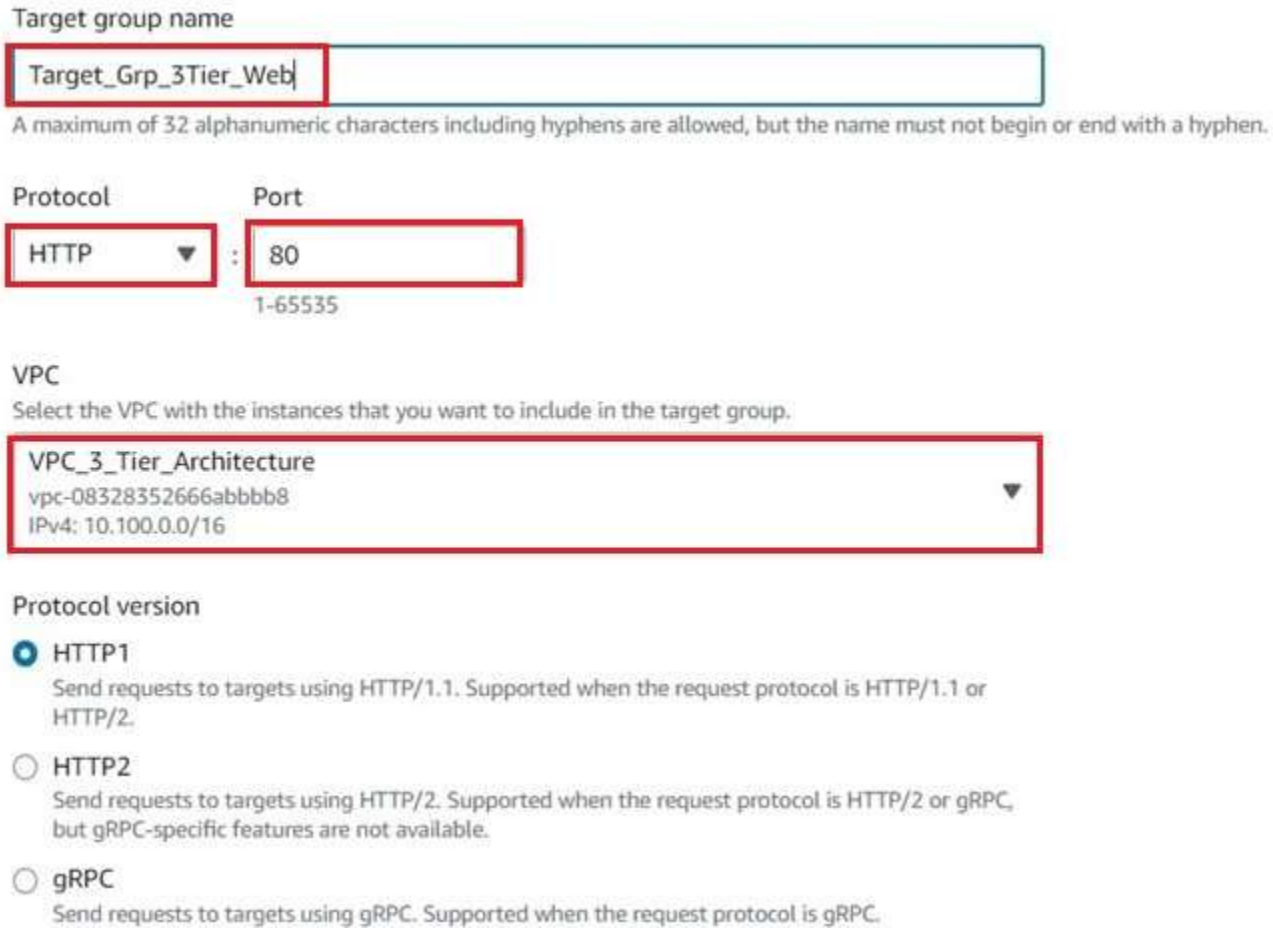
In the navigation pane, click on "Target Groups" under "Load Balancing."

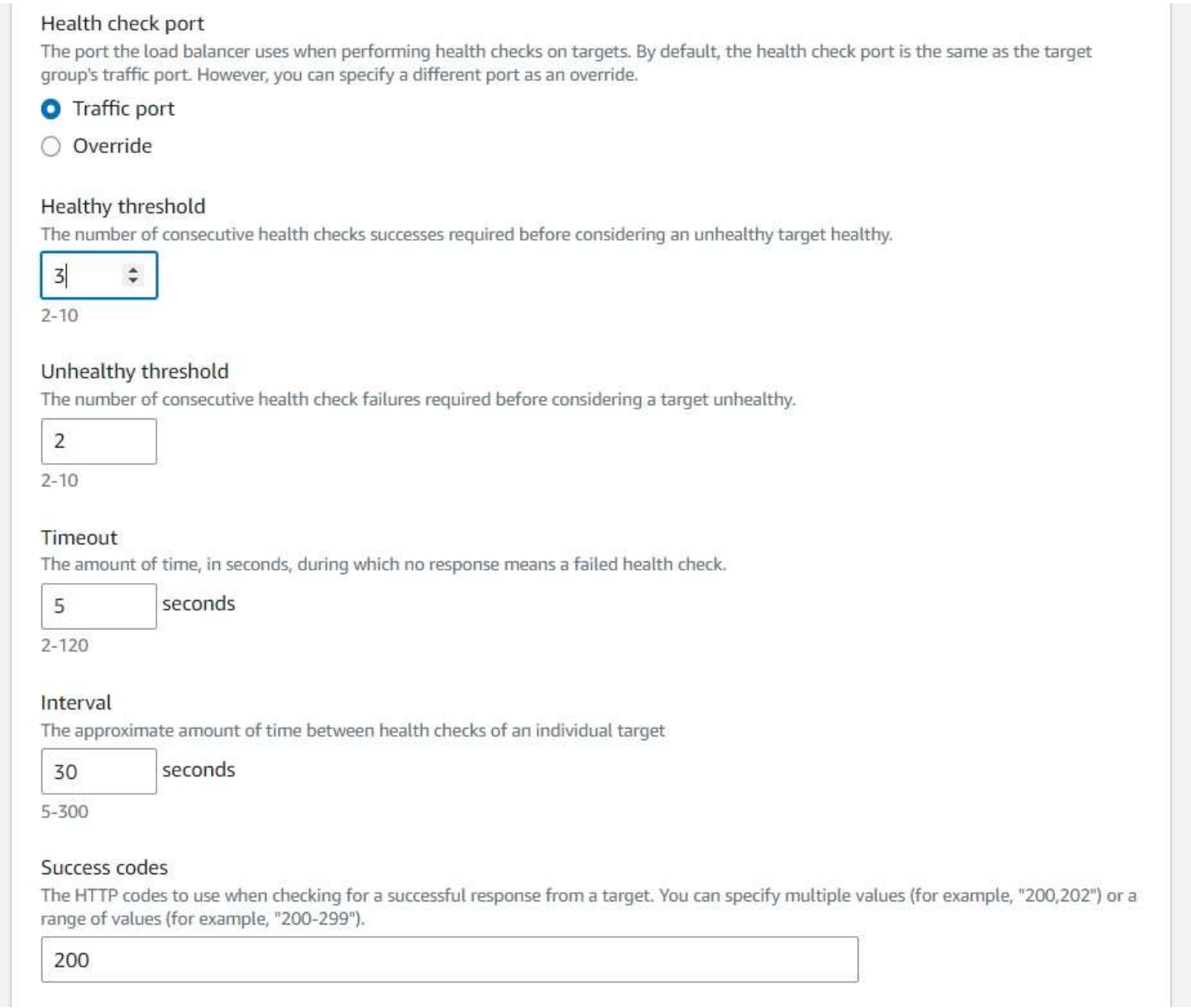
Click on the "Create target group" button and specify a name for the target group,

Configure the health checks for your target group. Health checks help the load balancer determine the availability of targets. Set the protocol, path, and interval for the health checks.

Specify the targets for your target group. This depends on the target type you selected earlier. If you choose "Instance," you can select one or more instances to include in the target group.

Once the target group is created, you can associate it with your Application Load Balancer to route traffic to the targets based on the specified rules.





STEP 9: CREATION OF APPLICATION LOAD BALANCERS (ALB)

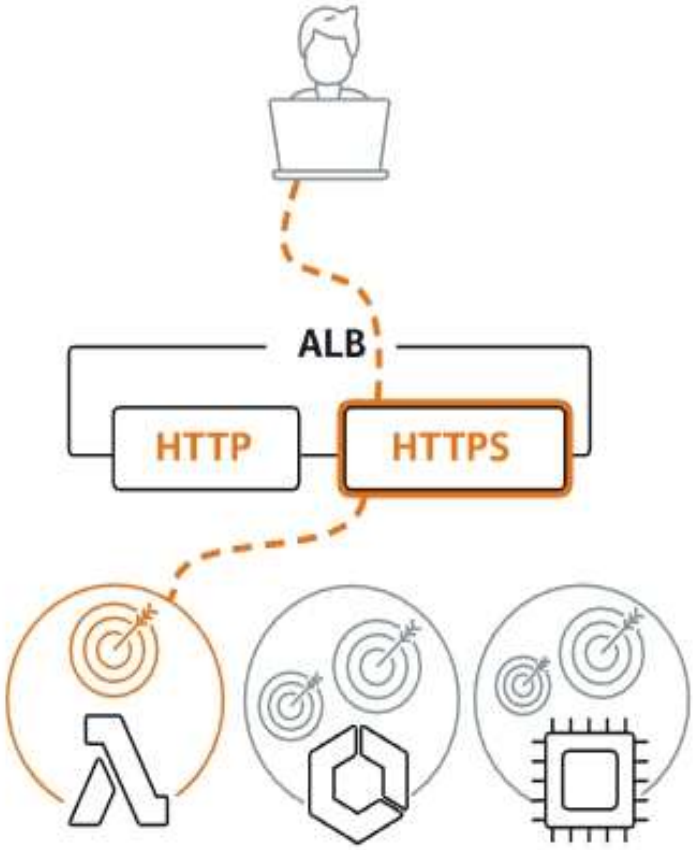
Now, our next task will be to create two (2) Application Load Balancers (ALB)s.

The first **ALB** will run at the web tier to add an **extra layer of security.**

The second **ALB** will run at the Application Tier and will act as a **communication bridge** between the **Web Tier** and the **Application tier.**

Now, let's proceed to create our Application Load Balancer for the Web Tier.

Application Load Balancer Info



Choose an Application Load Balancer when you need a flexible feature set for your applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers.

Create

Create Application Load Balancer Info

The Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets such as Amazon EC2 instances, microservices, and containers, based on request attributes. When the load balancer receives a connection request, it evaluates the listener rules in priority order to determine which rule to apply, and if applicable, it selects a target from the target group for the rule action.

▸ **How Elastic Load Balancing works**

Basic configuration

Load balancer name

Name must be unique within your AWS account and can't be changed after the load balancer is created.

3Tier-Web-ALB

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Scheme Info

Scheme can't be changed after the load balancer is created.

- (•) Internet-facing
  An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. Learn more
- ( ) Internal
  An internal load balancer routes requests from clients to targets using private IP addresses.

IP address type Info

Select the type of IP addresses that your subnets use.

- (•) IPv4
  Recommended for internal load balancers.
- ( ) Dualstack
  Includes IPv4 and IPv6 addresses.

Network mapping Info

The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

VPC Info

Select the virtual private cloud (VPC) for your targets or you can create a new VPC. Only VPCs with an internet gateway are enabled for selection. The selected VPC can't be changed after the load balancer is created. To confirm the VPC for your targets, view your target groups.

VPC_3_Tier_Architecture
vpc-08328352666abbbb8
IPv4: 10.100.0.0/16

Mappings Info

Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection.

[x] **us-east-1a (use1-az4)**

Subnet

subnet-0fa9e2c9bb2798784 Public_Sub_Web_1a

IPv4 address

Assigned by AWS

[x] **us-east-1b (use1-az6)**

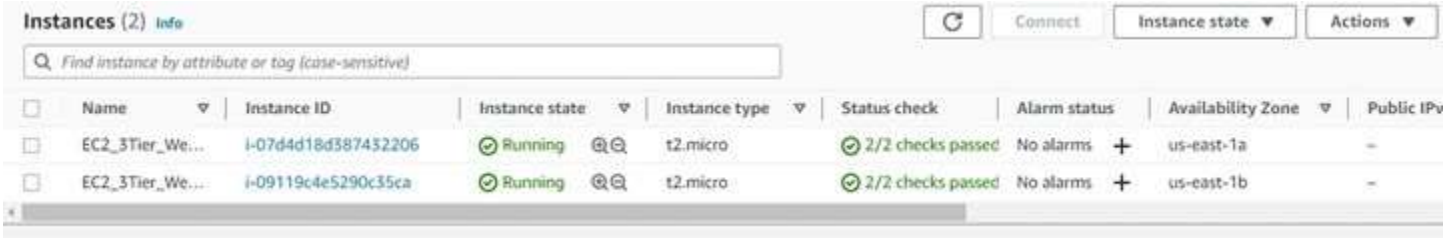
Subnet

subnet-020cdcfc46f5400f1 Public_Sub_Web_1b

My Load Balancer "3Tier-Web-ALB" was successfully created.

STEP 10: TESTING OF THE WEB TIER

Navigated to EC2 Dashboard, clicked on running instances and could see two EC2 instances deployed and running.



Copied the EC2 instance public IP addresses, pasted it on a web browser.

Yes, everything is working perfectly as shown above.

APPLICATION TIER

STEP 1: NETWORK CONFIGURATION - VPC CREATION

We do not need to create a new VPC at this point for the application tier, since we already had our **(VPC_3_Tier_Architecture)** created, that is currently in use for this project.

STEP 2: SUBNET CREATION:

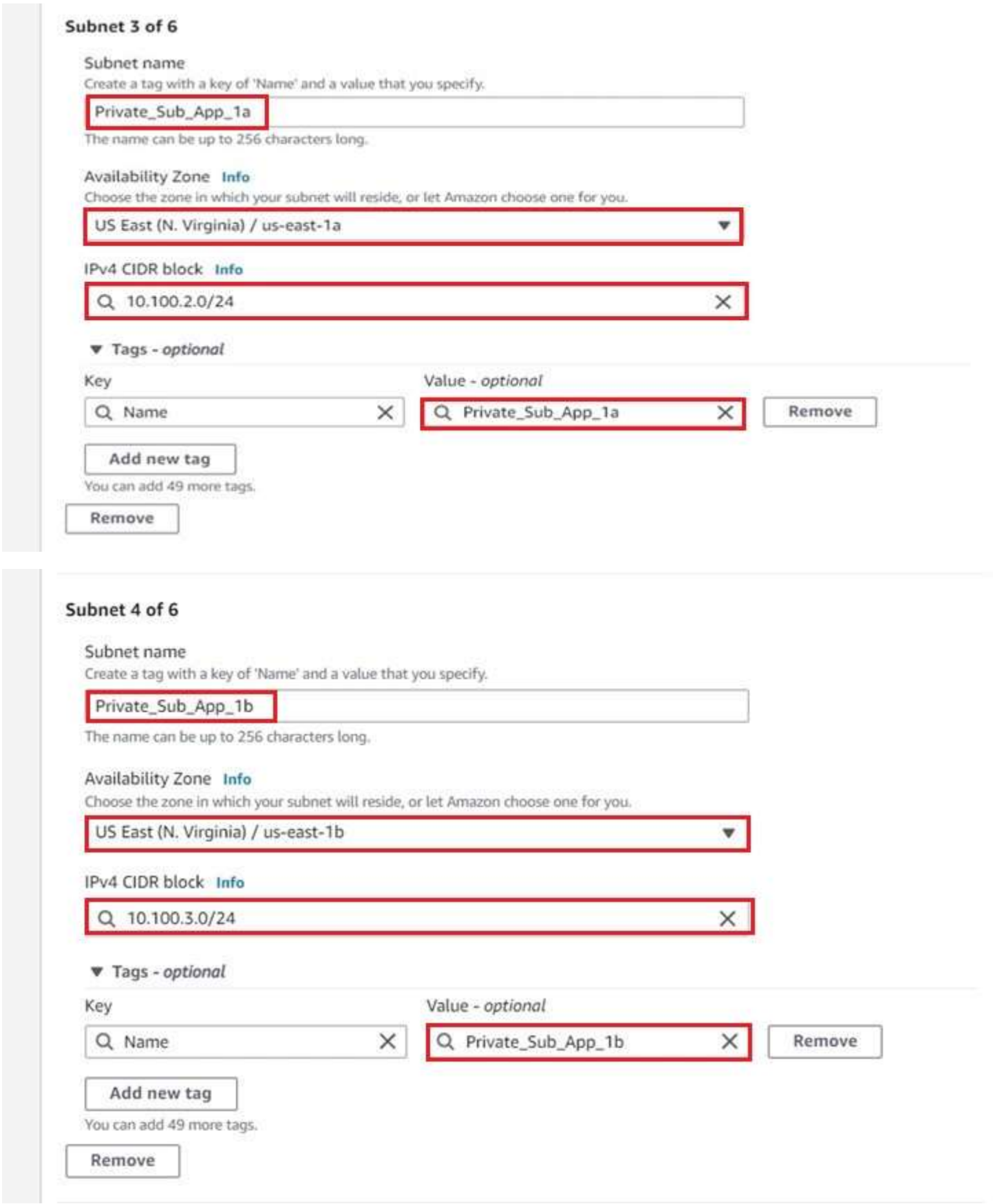
For the **Application Tier**, we already created two 2 private subnets:

1.**Private_Sub_App_1a**, with cidr **10.100.2.0/24**

2.**Private_Sub_App_1b**, with cidr **10.100.3.0/24**

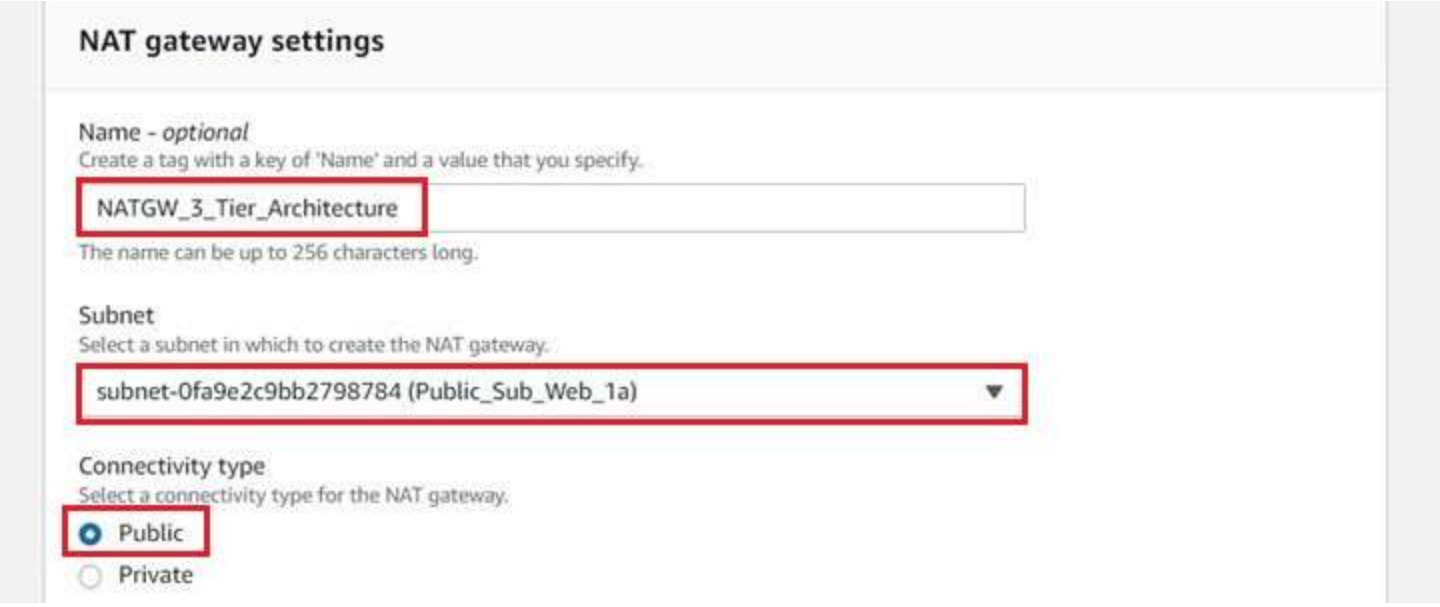
We currently do not have the need to create new Application Tier subnets at this point.

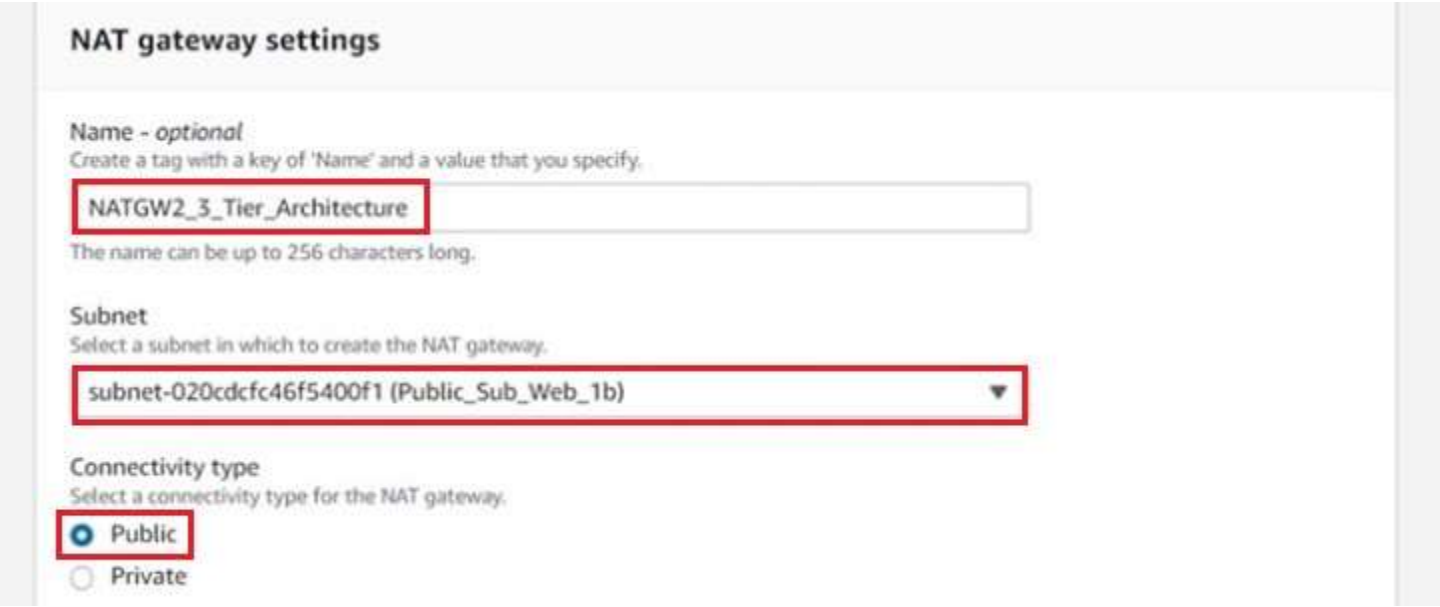
Recall that we initially created 2 private subnets for the Application Tier, alongside 2 private subnets for the Database Tier and 2 public subnets for Web Tier respectively during the initial Web tier subnet creation stage of this project.



STEP 4: NAT GATEWAY CREATION:

We do not need to create a new **NAT gateway** at this point, this is because we already had two NAT gateways **(NATGW_3_Tier_Architecture)** and **(NATGW2_3_Tier_Architecture)** created in the Web Tier of this architecture, that is currently in use for this project.





STEP 5: ELASTIC IP ADDERESS

Elastic IP **(eipalloc-0f039f4d75d2e552b)** was allocated to my first **NAT gateway (NATGW_3_Tier_Architecture)** that was already created.

So, we do not need to request another elastic IP address at this point.

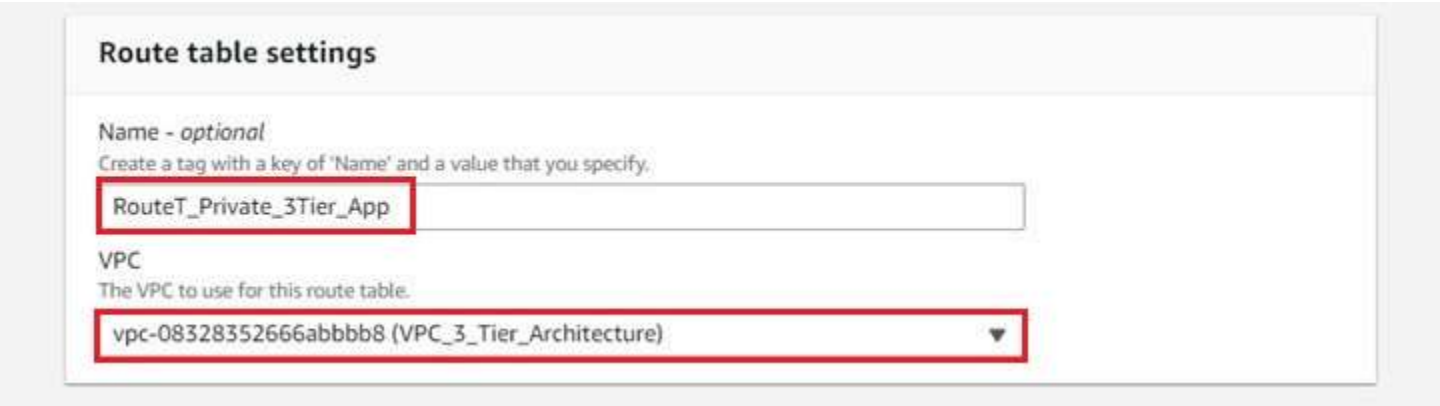
Elastic IP **(eipalloc-03a435d42634fb1c8)** was allocated to my second **NAT gateway (NATGW2_3_Tier_Architecture)** that was already created.

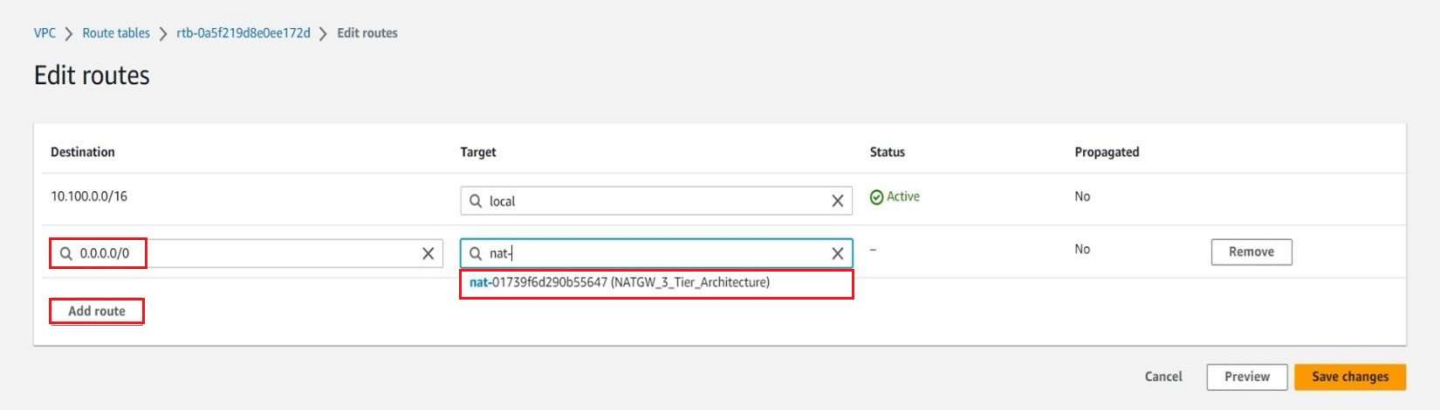
So, we do not need to request another elastic IP address at this point neither.

STEP 6: CREATE AND EDIT ROUTE TABLE

For the Application Tier, the route table **(RouteT_Private_3Tier_App)** was already created and edited to route traffic from the **Application Tier private subnet (Private_Sub_App_1a)** and from the **Database Tier private subnet (Private_Sub_DB_1a)** to NAT gateway **(NATGW_3_Tier_Architecture)** that resides in the Web Tier public subnet (Public_Sub_Web_1a).

In other words, the Route Table named **(RouteT_Private_3Tier_App)** consumes traffic that was routed to it from the Application Tier private subnet **(Private_Sub_App_1a)** and from the Database Tier private subnet **(Private_Sub_DB_1a)** and then wires these traffic to NAT gateway **(NATGW_3_Tier_Architecture)** that resides in the Web Tier public subnet **(Public_Sub_Web_1a).** So, we currently do not need to create new route tables at this point.





STEP 7: EDIT SUBNET ASSOCIATION

This task was completed during the Web Tier stage of this project, so there is nothing to do here.

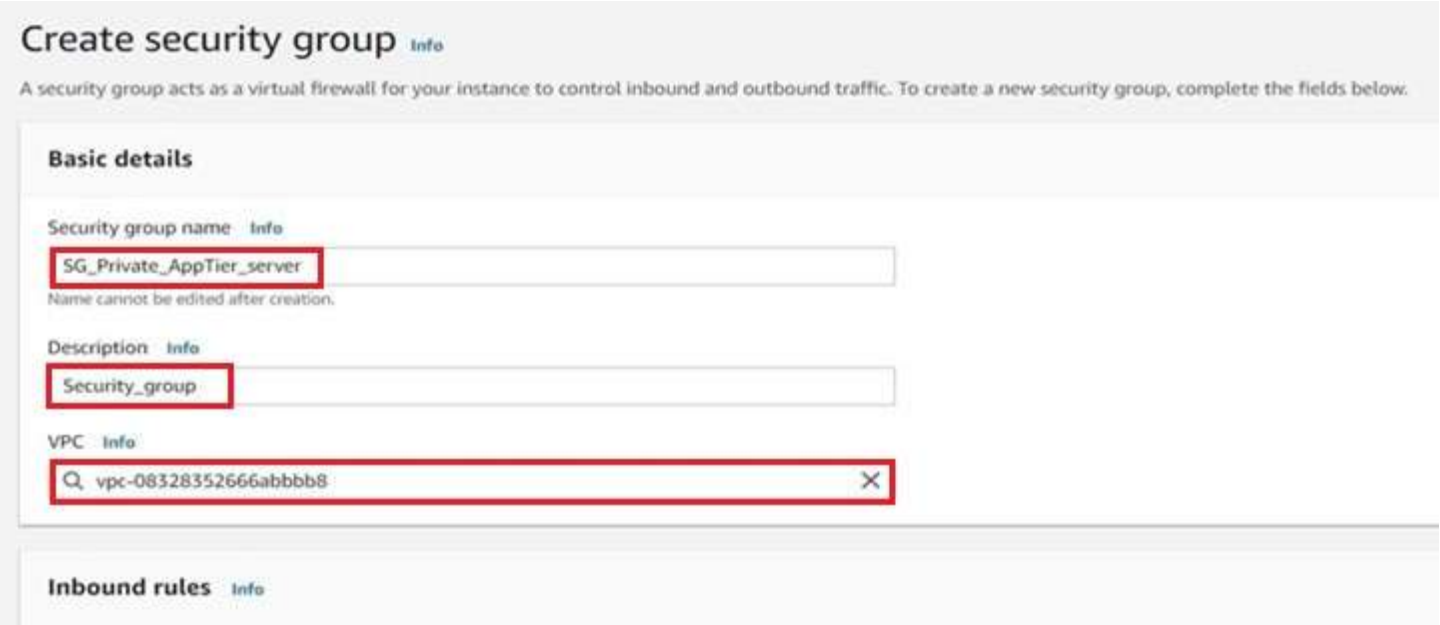
The Application Tier Route Table **(RouteT_Private_3Tier_App)** is associated with private subnet **(Private_Sub_App_1a)** and with Database Tier subnet **(Private_Sub_DB_1a)** in this my case.

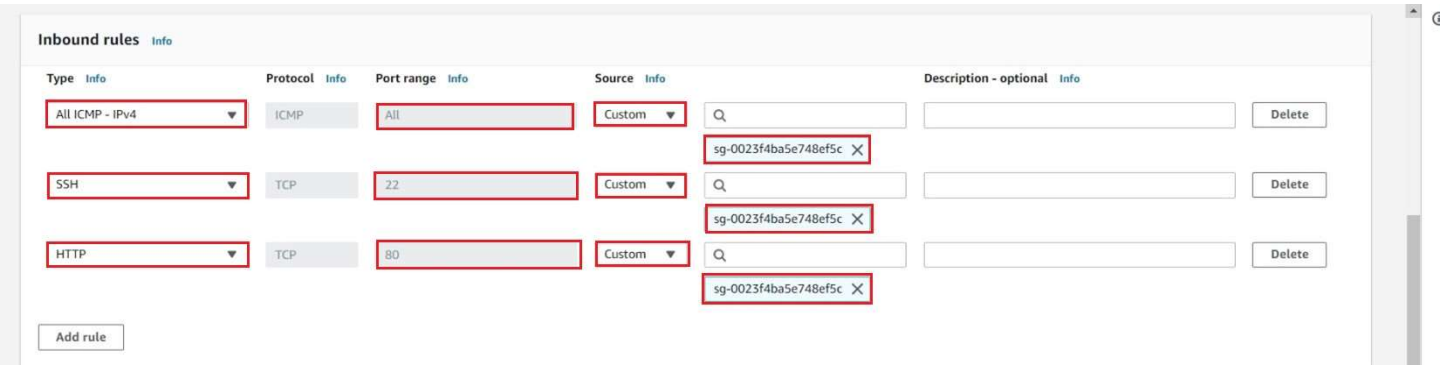
STEP 8: SECURITY GROUP

Let's navigate to the EC2 dashboard, scroll down to select security groups, click on create security group, give it a name, **(SG_Private_AppTier_server)** in my case, add description, and specify your VPC.

Choose the security group rule that will control our web server inbound and outbound traffic.

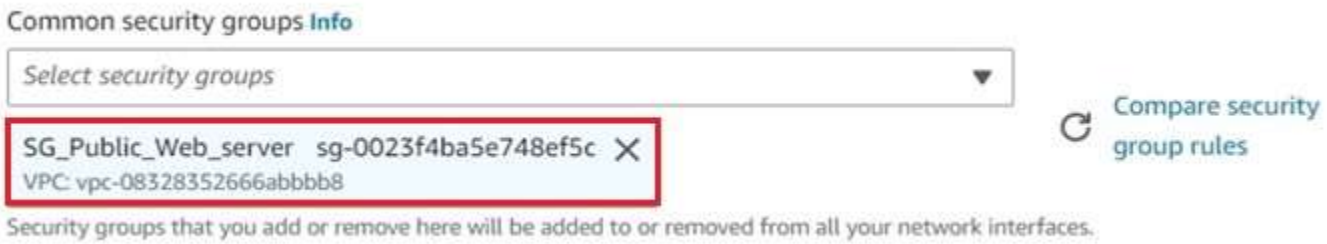
We must configure the **Application Server** security group to allow inbound permission from the Web Tier security group as the source of incoming traffic. That is, our Application Tier security group **(SG_Private_AppTier_server)** should be referencing or pointing towards the Web Server security group **(SG_Public_Web_server = sg-00223f4ba5e748ef5c)** as it's source of incoming traffic.





The **SG_private_AppTier_server** security group inbound rule was successfully referenced to the **SG_Public_Web_server = sg-00223f4ba5e748ef5c** as its traffic source.

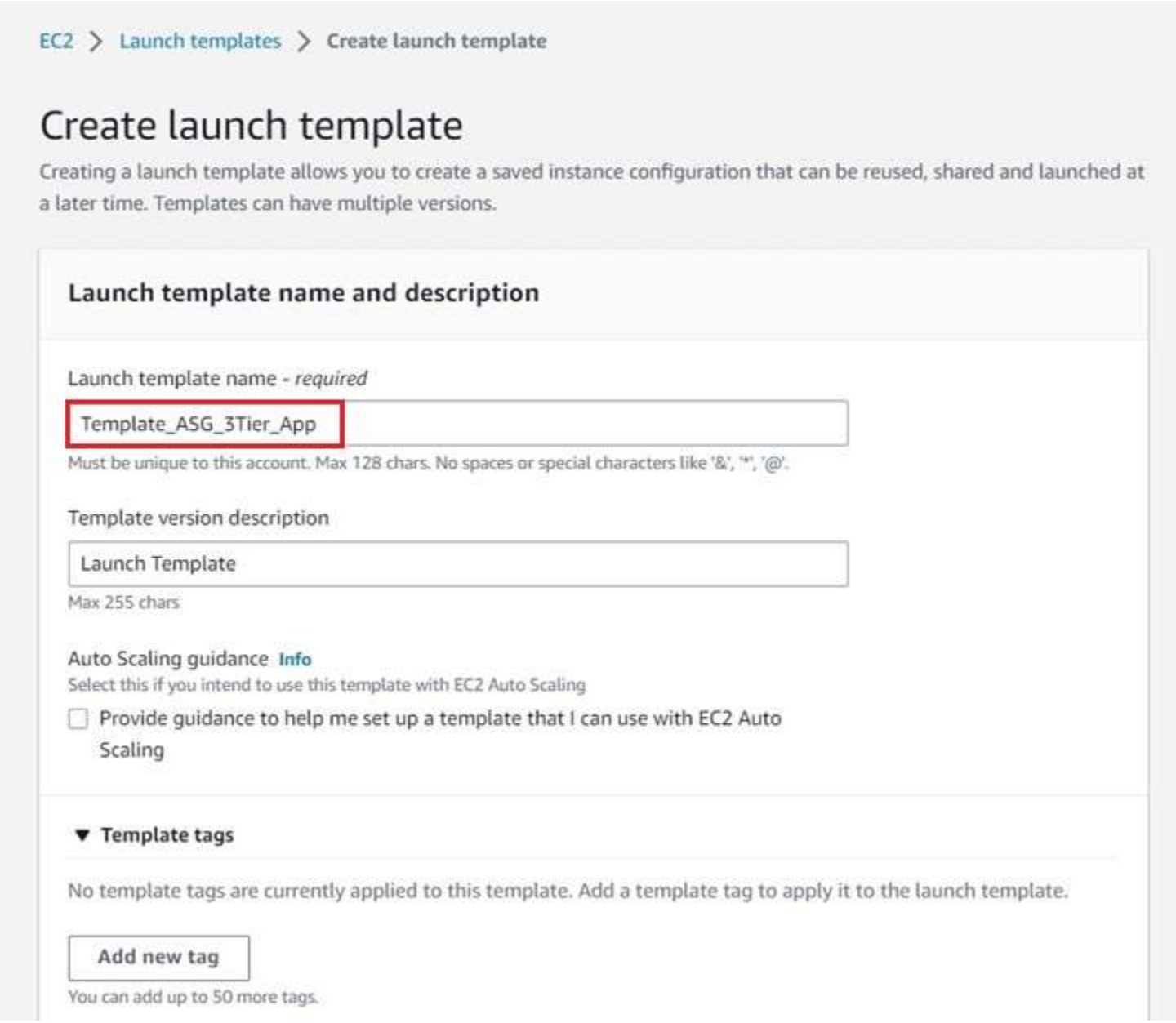
Below is a screenshot of my Web Server security (**SG_Public_Web_server)** only for reference.



Security groups are stateful by default, which means it can keep track of the state of network traffic that flows in and out of EC2 instances, and it makes intelligent decisions about allowing or blocking traffic to EC2 instances.

STEP 9: CREATION OF LAUNCH TEMPLATE

Following the same steps from the Web Tier, let's create a launch template using the same AMI, t2. micro, and Key pair. The Source for each protocol would be the Webserver security group ((**SG_Public_Web_server)**



Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Recents | Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | S

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type — Free tier eligible

ami-0bef6cc322bfff646 (64-bit (x86)) / ami-09212035c6444f37a (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2 Kernel 5.10 AMI 2.0.20230515.0 x86_64 HVM gp2

Architecture AMI ID

Instance type Info

Advanced

Instance type

t2.micro — Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true

On-Demand Windows pricing: 0.0162 USD per Hour

On-Demand SUSE pricing: 0.0116 USD per Hour

On-Demand RHEL pricing: 0.0716 USD per Hour

On-Demand Linux pricing: 0.0116 USD per Hour

All generations

Compare instance types

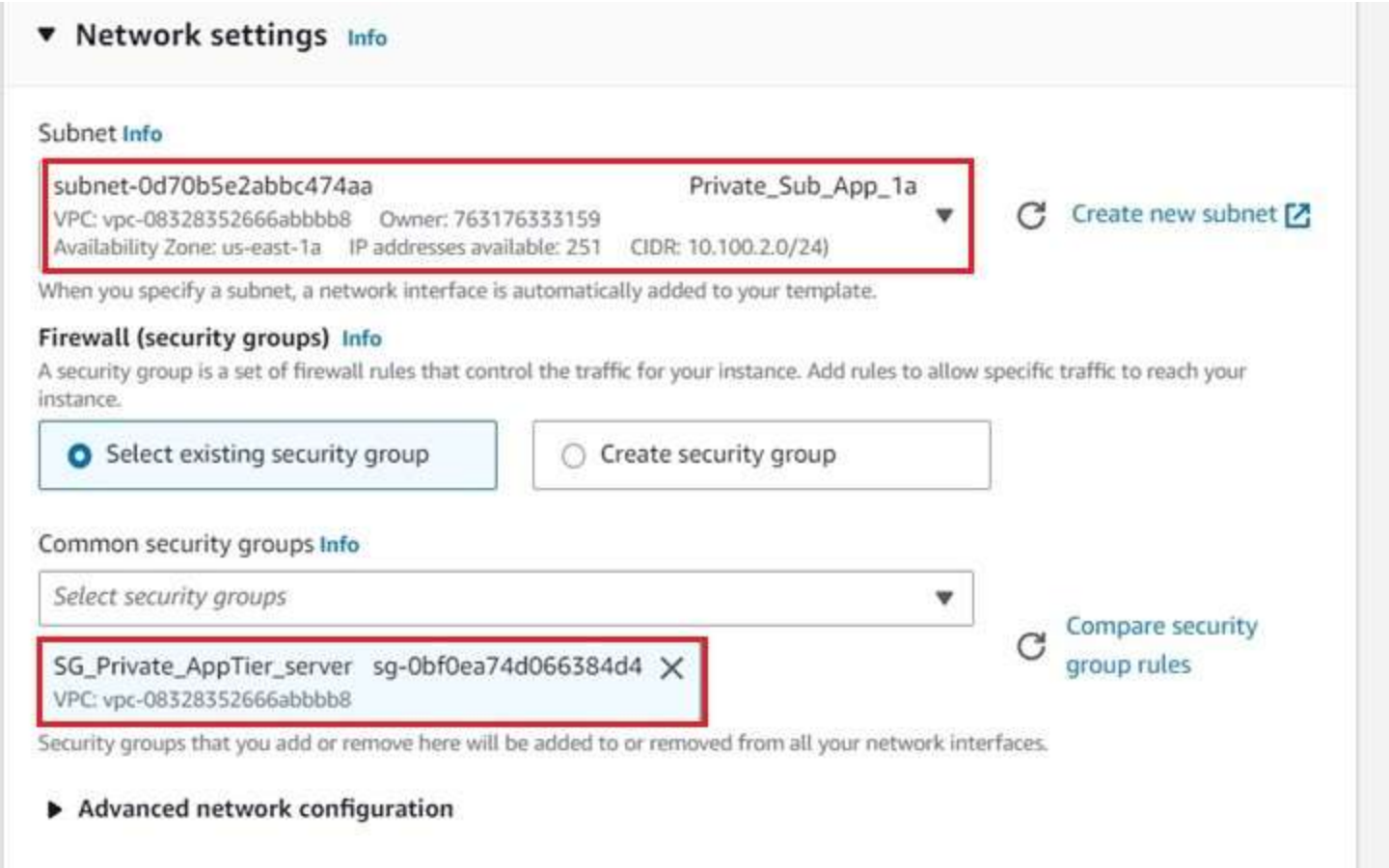
Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name

3Tier_KPair

Create new key pair



STEP 10: CREATE AN AUTO SCALING GROUP (ASG)

Now that the launch template is created, let's configure an **auto scaling group** (ASG) for the private Application Tier servers.

Navigate to Autoscaling groups, click on create, specify a name **(3Tier_App_ASG),** select the launch template that we just created **(Template_ASG_3Tier_App**), specify the VPC and availability zones/subnets, then specify the 2 private subnets **(Private_Sub_App_1a and Private_Sub_App_1b).**

Name

Auto Scaling group name
Enter a name to identify the group.

3Tier_Web_ASG

Must be unique to this account in the current Region and no more than 255 characters.

Launch template Info

Switch to launch configuration

Launch template
Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.

Template_ASG_3Tier_App

Create a launch template

Version

Default (1)

Create a launch template version

Description	Launch template	Instance type

Launch template Info

Switch to launch configuration

Launch template
Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.

Template_ASG_3Tier_App

Create a launch template

Version

Default (1)

Create a launch template version

Description	Launch template	Instance type
Launch Template	Template_ASG_3Tier_App lt-063c3c8b42fcd9d09	t2.micro
AMI ID	**Security groups**	**Request Spot Instances**
ami-0bef6cc322bfff646	-	No
Key pair name	**Security group IDs**	
3Tier_KPair	sg-0bf0ea74d066384d4	

Additional details

Storage (volumes)	Date created
-	Fri May 26 2023 15:14:05 GMT-0500 (Central Daylight Time)

Cancel Next

Network Info

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

VPC

Choose the VPC that defines the virtual network for your Auto Scaling group.

vpc-08328352666abbbb8 (VPC_3_Tier_Architec...
10.100.0.0/16

Create a VPC

Availability Zones and subnets

Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.

Select Availability Zones and subnets

us-east-1a | subnet-0d70b5e2abbc474aa (Private_Sub_App_1a)
10.100.2.0/24

us-east-1b | subnet-07ae4238dd6b29cd6 (Private_Sub_App_1b)
10.100.3.0/24

Create a subnet

Configure group size and scaling policies - *optional* Info

Set the desired, minimum, and maximum capacity of your Auto Scaling group. You can optionally add a scaling policy to dynamically scale the number of instances in the group.

Group size - *optional* Info

Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum capacity limits. Your desired capacity must be within the limit range.

Desired capacity

2

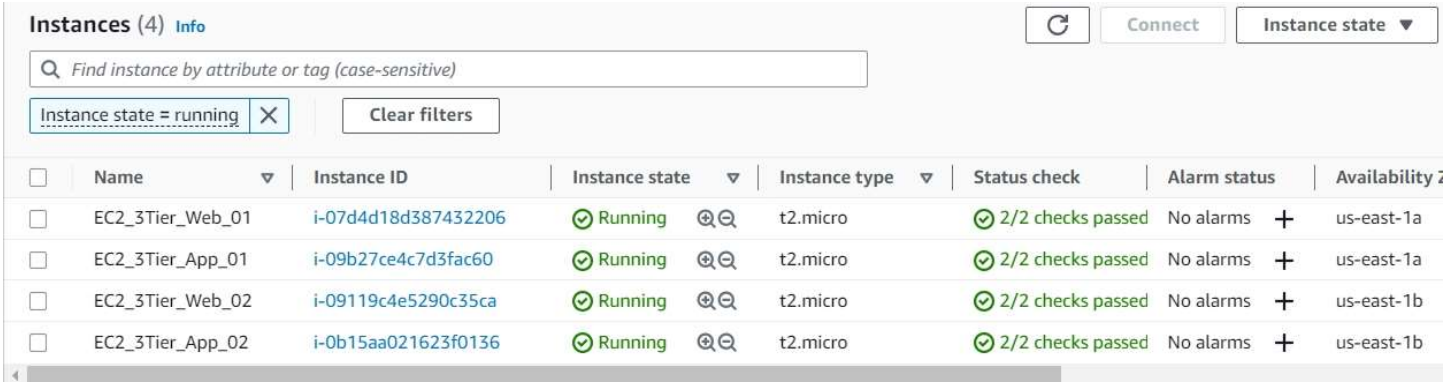
Minimum capacity

2

Maximum capacity

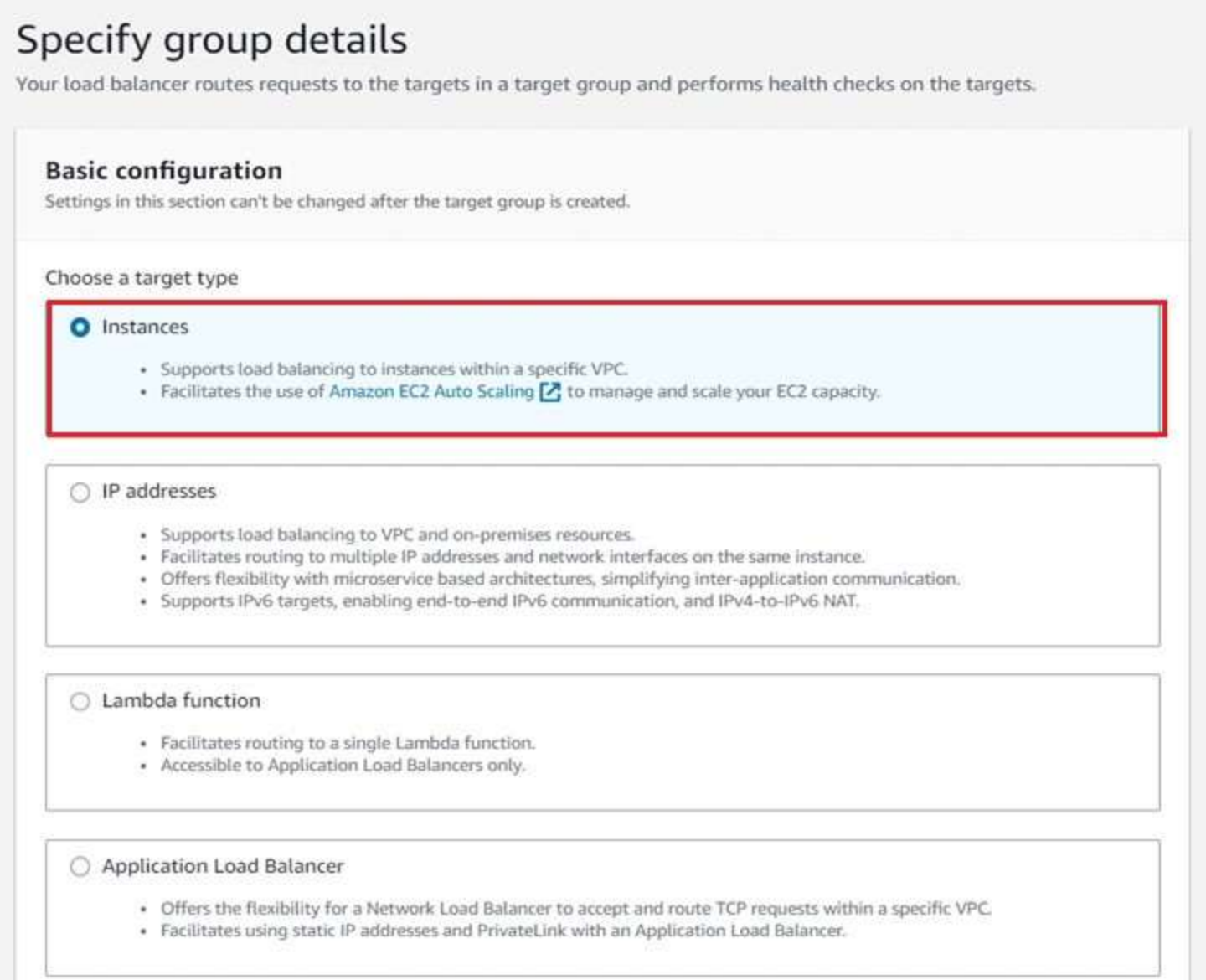
3

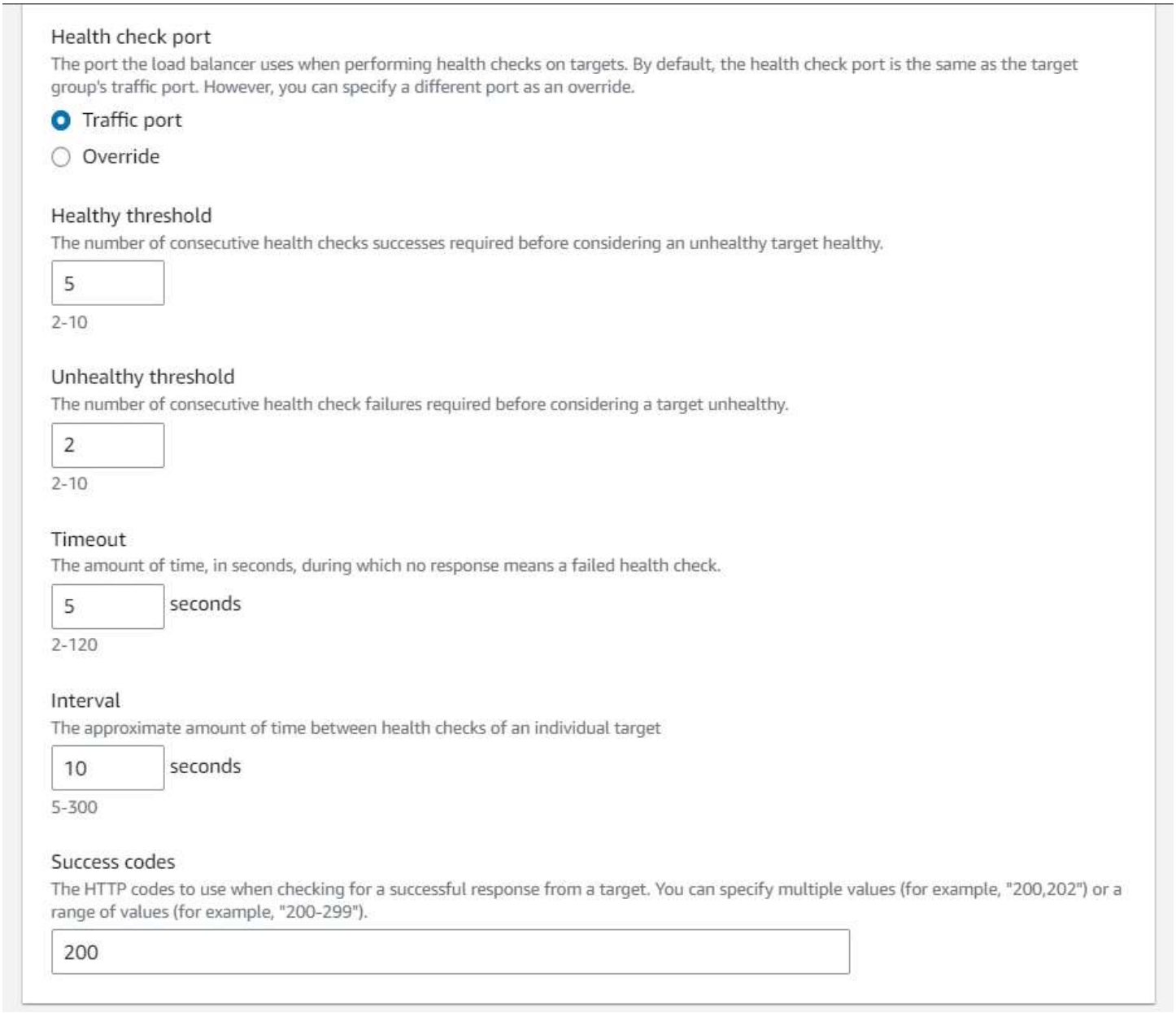
Navigate to EC2 dashboard, select running Instances and we noticed that Auto Scaling has created two additional EC2 instances that are currently running in the private subnet.



STEP 11: CREATE TARGET GROUP

Now, let's proceed to create a target group that would be used to launch our ALB.





Target group (Target-Grp-3Tier-App) was successfully created.

STEP 12: CREATE APPLICATION LOAD BALANCER (ALB)

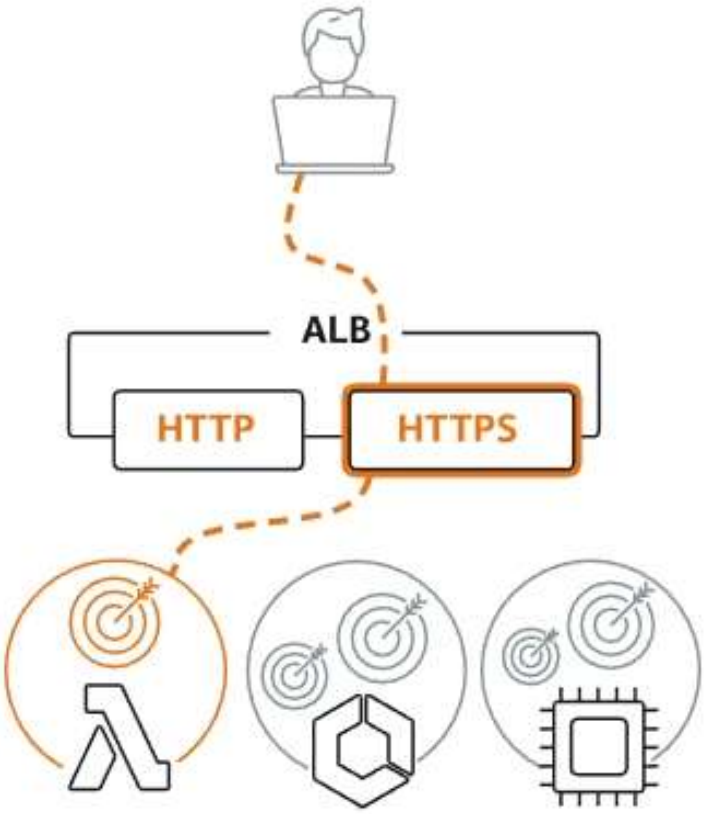
IMPORTANT: Load balancer scheme should be configured as "Internet-facing"

IMPORTANT: Default security group should be selected, for the Application Tier security group to allow inbound permission from the Web Tier security group as the source of incoming traffic.

IMPORTANT: Listeners and Routing. Default routing will be forwarded to a new Target group **(Target-Grp-3Tier-App)**

Load balancer types

Application Load Balancer Info



Choose an Application Load Balancer when you need a flexible feature set for your applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers.

Create

Basic configuration

Load balancer name

Name must be unique within your AWS account and can't be changed after the load balancer is created.

3Tier-App-ALB

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Scheme Info

Scheme can't be changed after the load balancer is created.

- (•) Internet-facing
  An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. Learn more
- ( ) Internal
  An internal load balancer routes requests from clients to targets using private IP addresses.

IP address type Info

Select the type of IP addresses that your subnets use.

- (•) IPv4
  Recommended for internal load balancers.
- ( ) Dualstack
  Includes IPv4 and IPv6 addresses.

Subnet

subnet-0d70b5e2abbc474aa Private_Sub_App_1a

⚠ The selected subnet does not have a route to an internet gateway. This means that your load balancer will not receive internet traffic.
You can proceed with this selection; however, for internet traffic to reach your load balancer, you must update the subnet's route table in the VPC console.

IPv4 address

Assigned by AWS

[x] **us-east-1b (use1-az6)**

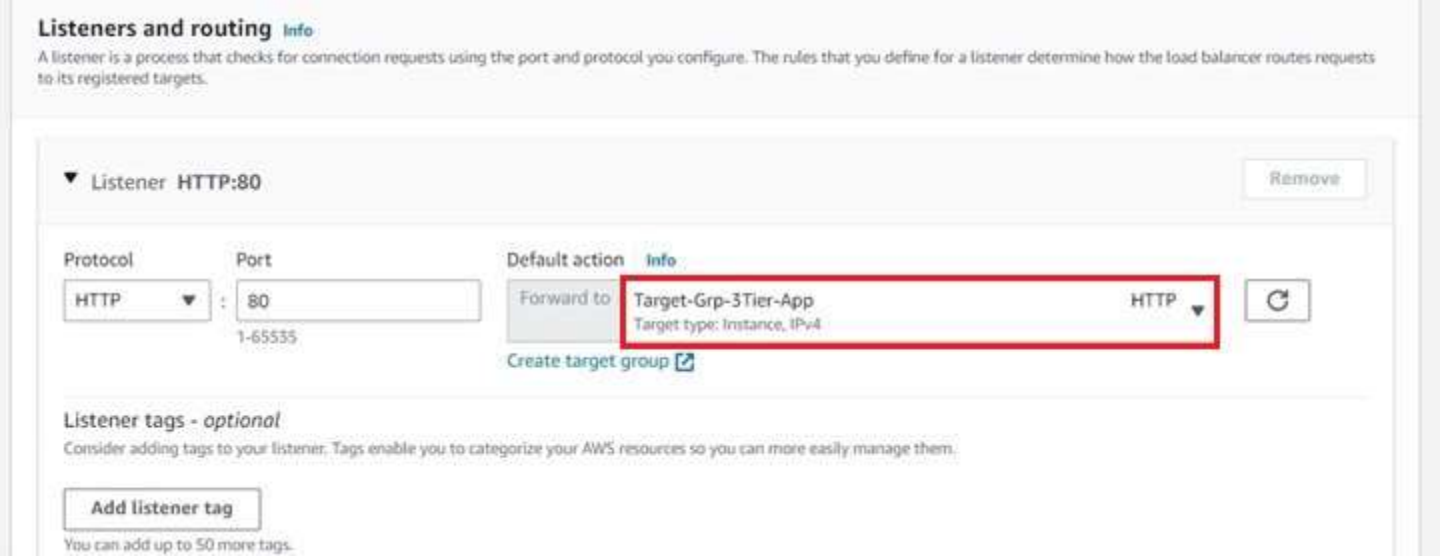
Subnet

subnet-07ae4238dd6b29cd6 Private_Sub_App_1b

⚠ The selected subnet does not have a route to an internet gateway. This means that your load balancer will not receive internet traffic.
You can proceed with this selection; however, for internet traffic to reach your load balancer, you must update the subnet's route table in the VPC console.

IPv4 address

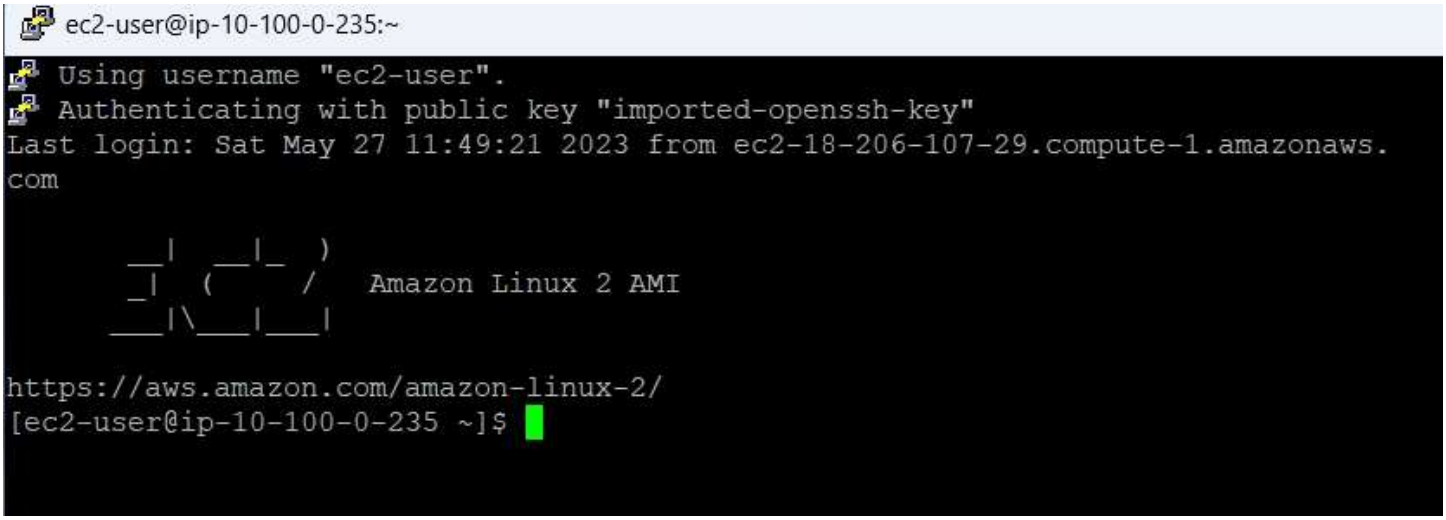
Assigned by AWS



The Application Load Balancer was successfully created, and it is currently up and running.

STEP 12: CONNECT TO THE EC2 MACHINES

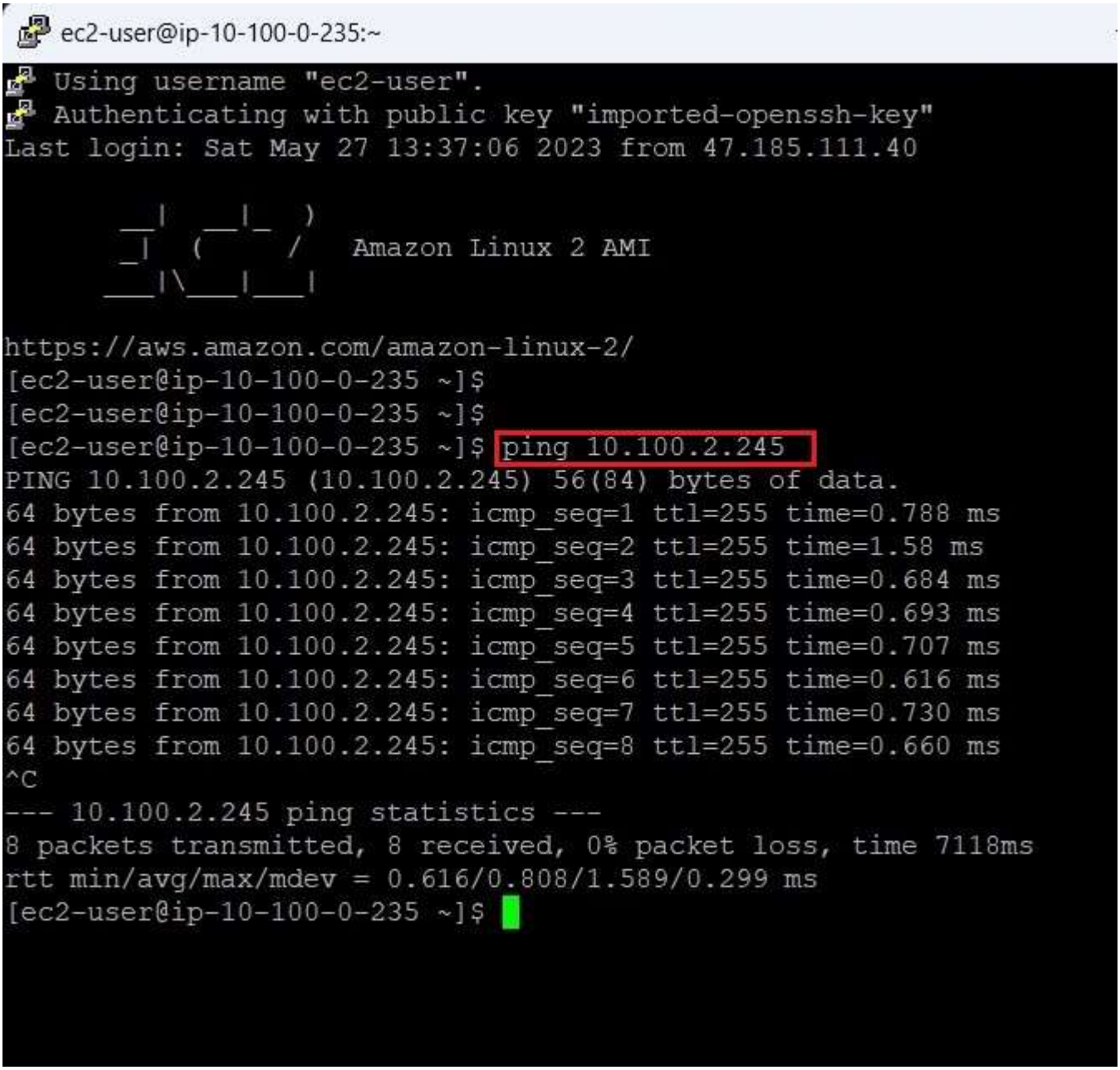
Now, we need to connect to the EC2 instance in our public the Web Tier, I am currently using a Windows 11 machine, so I will be using Putty and 3Tier_KPair.ppk to connect.



Excellent! Connection was successfully established.

Note: I used puttygen to convert my .pem key to .ppk format. This is because putty only accepts private keys in the .ppk format when establishing a connection from Windows to Linux machines. But will accept private keys in the .pem format when connecting from Linux to Linux.

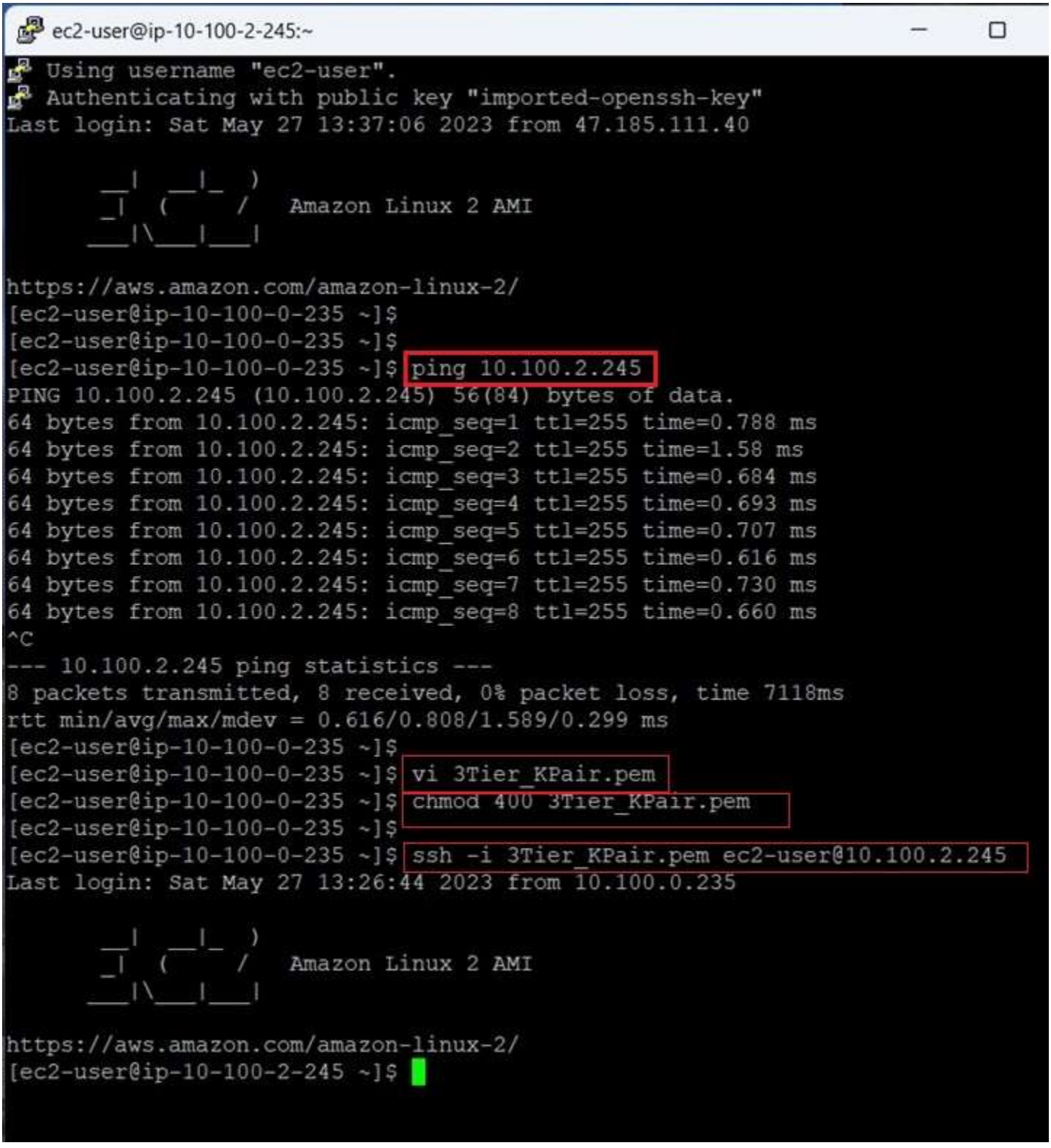
Next step will be to check if we can access the EC2 instance in the Private Application Tier from the public Web Tier by running the ping command.



Awesome! Pinging the EC2 in the private Application Tier subnet from the public Web Tier subnet was successful.

Now that ping was successful, our next task will be to connect from the public web tier EC2 instance to the private application tier EC2 instance using my 3Tier_KPair.pem key.

First, let's change the permissions of our 3Tier_KPair.pem by running chmod 400 3Tier_KPair.pem before connecting to the private Application Tier EC2 server.



Everything is working great as expected!

DATABASE TIER

NETWORK CONFIGURATION

The principal objective here is to configure the Database tier, and check connectivity to the Application Tier.

STEP 1: VPC CONFIGURATION

We do not need to create a new VPC at this point, since we already had our **(VPC_3_Tier_Architecture)** created, that is currently in use for this project.

STEP 2: SUBNET CREATION

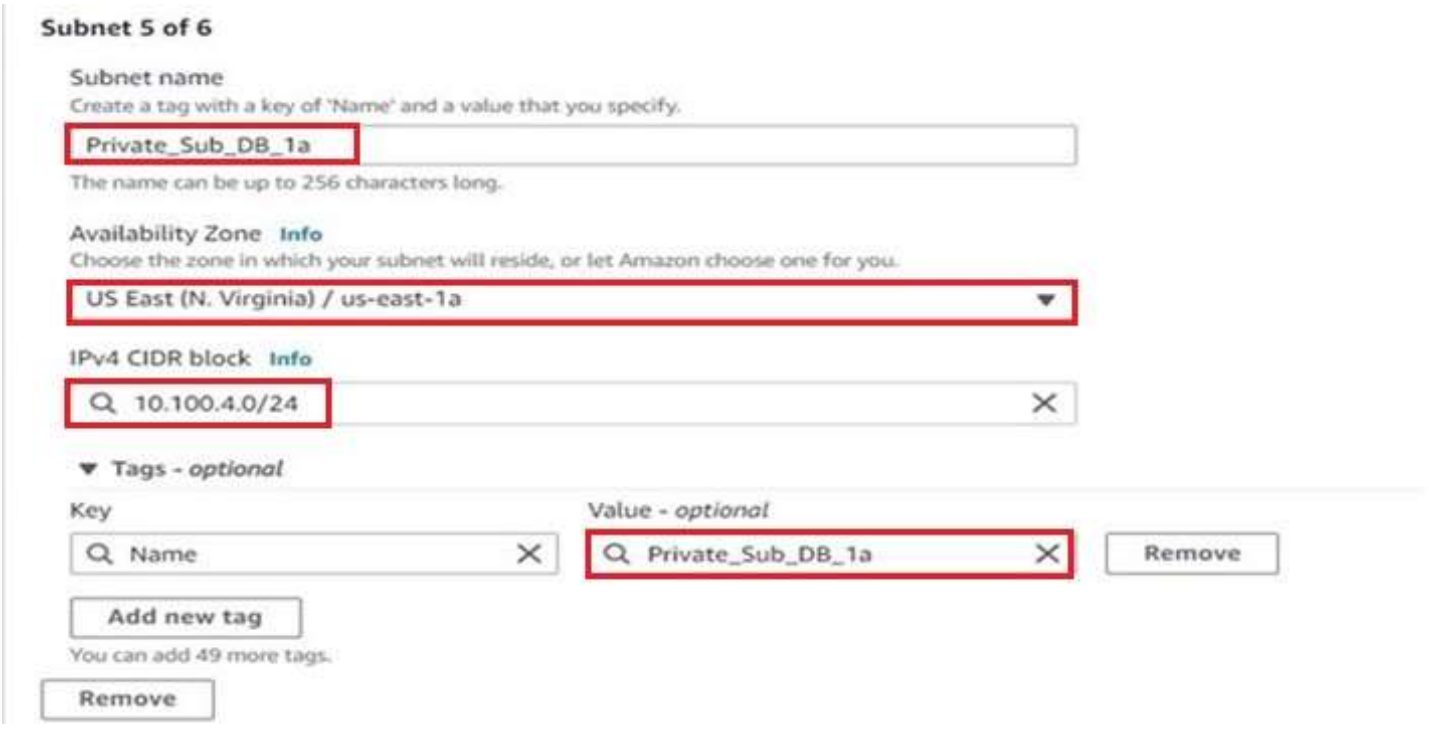
For the **Database Tier**, we already created two 2 private subnets:

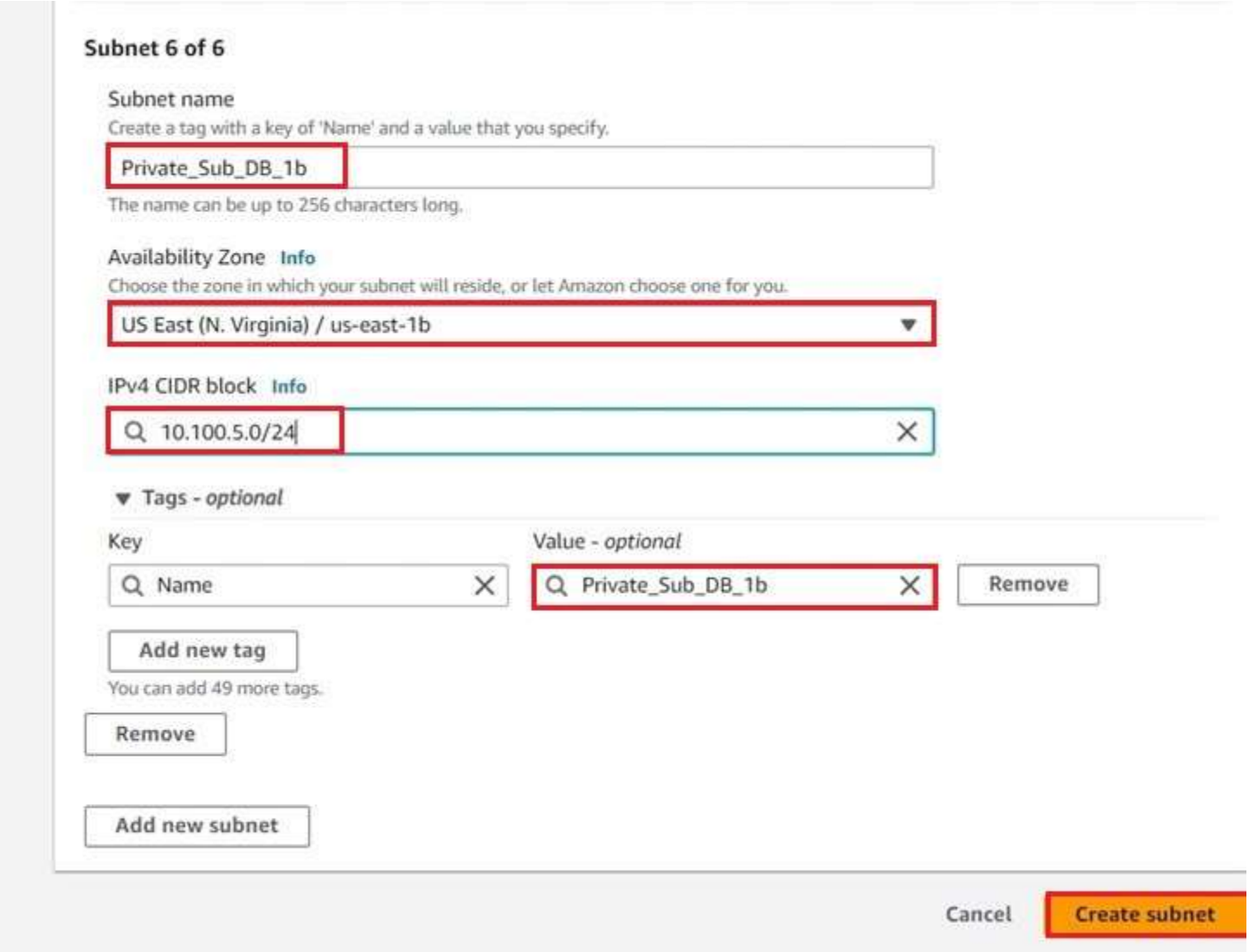
1.**Private_Sub_DB_1a**, with cidr **10.100.2.0/24**

2.**Private_Sub_DB_1b**, with cidr **10.100.3.0/24**

We currently do not have the need to create new Application Tier subnets at this point.

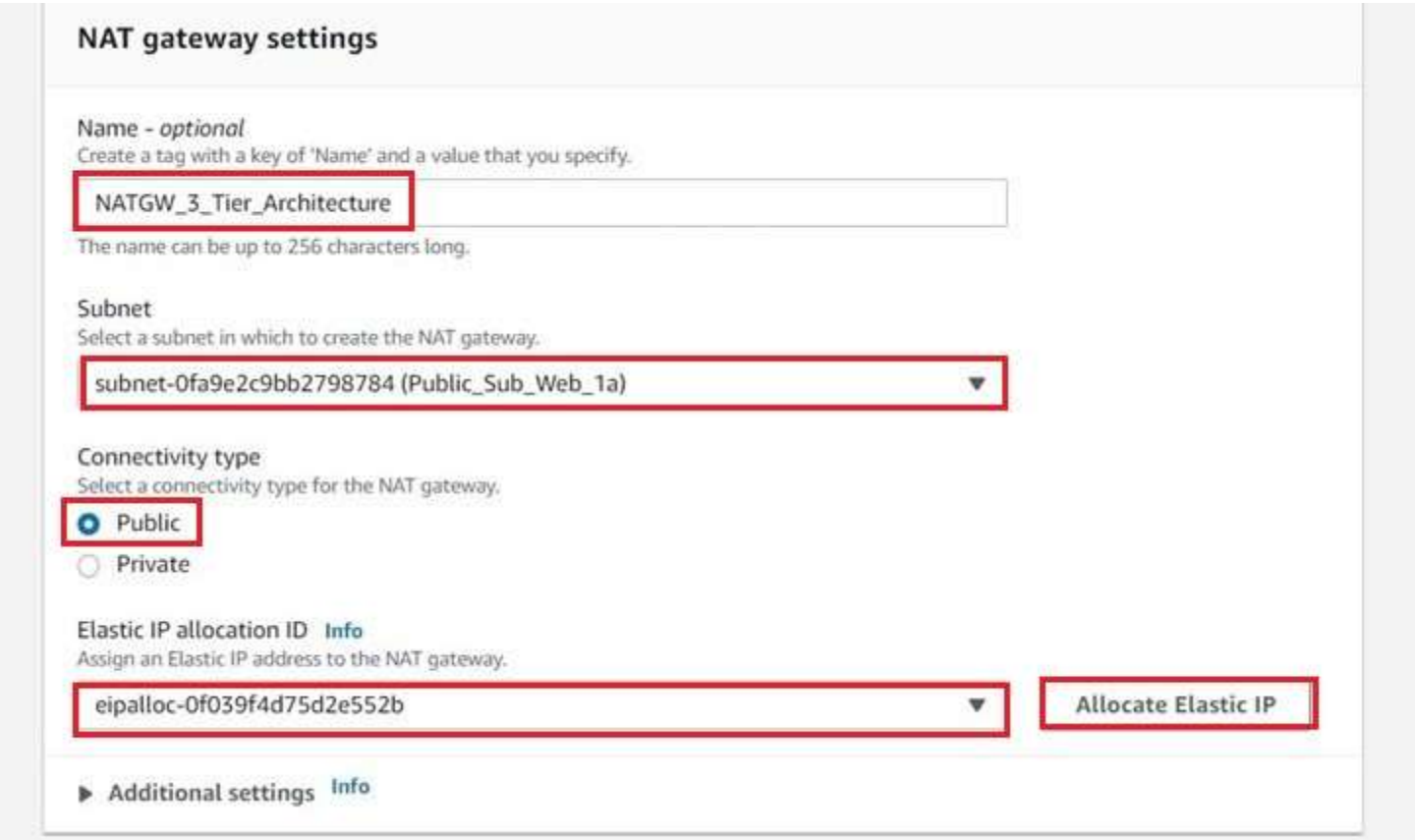
Recall that we initially created 2 private subnets for the Database Tier, alongside 2 private subnets for the Application Tier and 2 public subnets for Web Tier respectively during the initial Web tier subnet creation stage of this project.



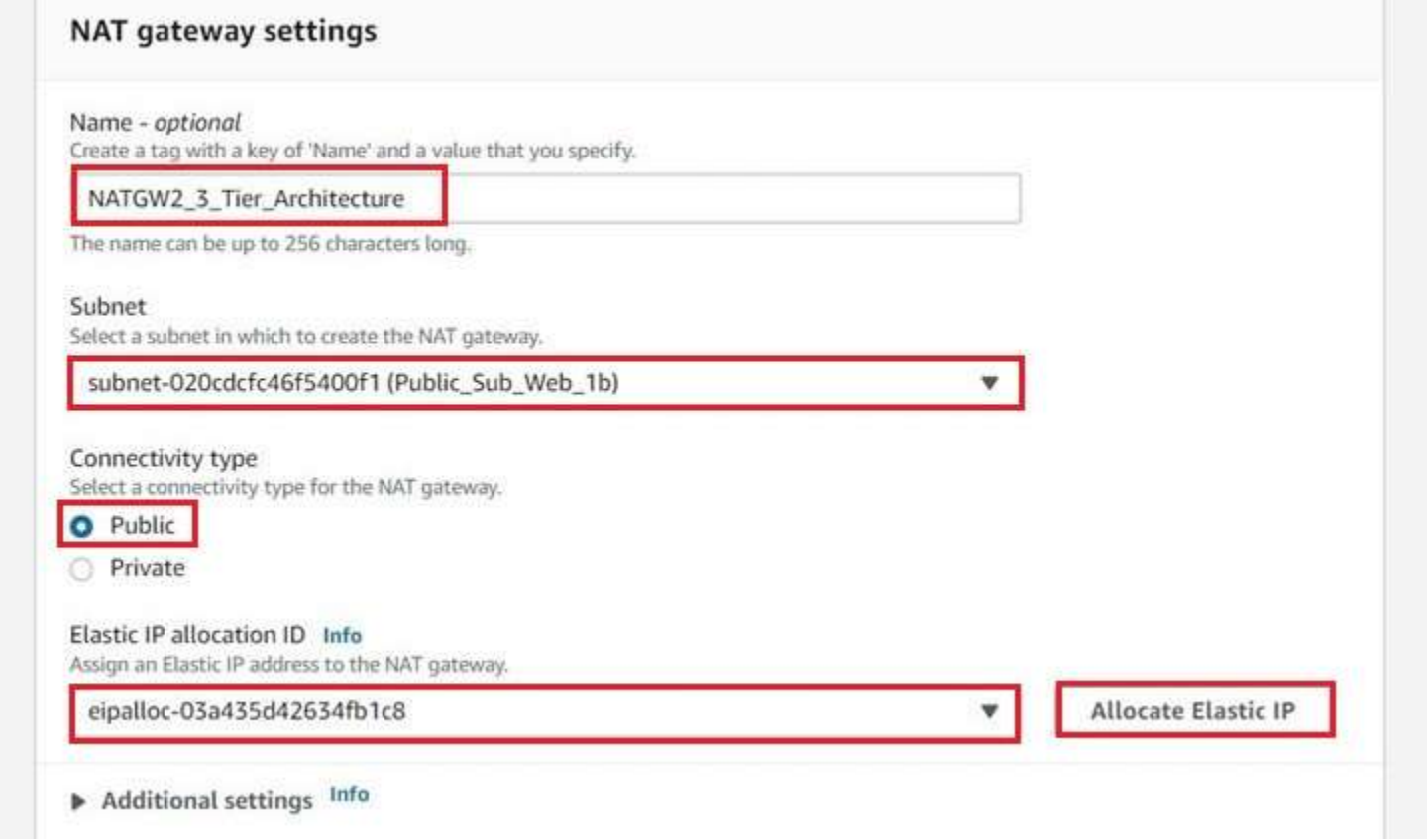


STEP 3: NAT GATEWAY CREATION:

We do not need to create a new NAT gateway at this point, this is because we already had two NAT gateways **(NATGW_3_Tier_Architecture)** and **(NATGW2_3_Tier_Architecture**) created in the Web Tier of this architecture, that is currently in use for this project.



The first NAT gateway **(NATGW_3_Tier_Architecture)** was created in the public Web subnet **(Public_Sub_Web_1a)**



The second NAT gateway **(NATGW2_3_Tier_Architecture)** was created in the public Web subnet **(Public_Sub_Web_1b)**

STEP 4: ELASTIC IP

Two elastic IP addresses **(eipalloc-0f039f4d75d2e552b**) and **(eipalloc-03a435d42634fb1c8)** were allocated during the creation of the NAT gateways, so we do not require another elastic IP address at this point neither.

Elastic IP **(eipalloc-0f039f4d75d2e552b)** was allocated to my first **NAT gateway (NATGW_3_Tier_Architecture)** that was already created.

So, we do not need to request another elastic IP address at this point.

Elastic IP **(eipalloc-03a435d42634fb1c8)** was allocated to my second **NAT gateway (NATGW2_3_Tier_Architecture)** that was already created.

So, we do not need to request another elastic IP address at this point either.

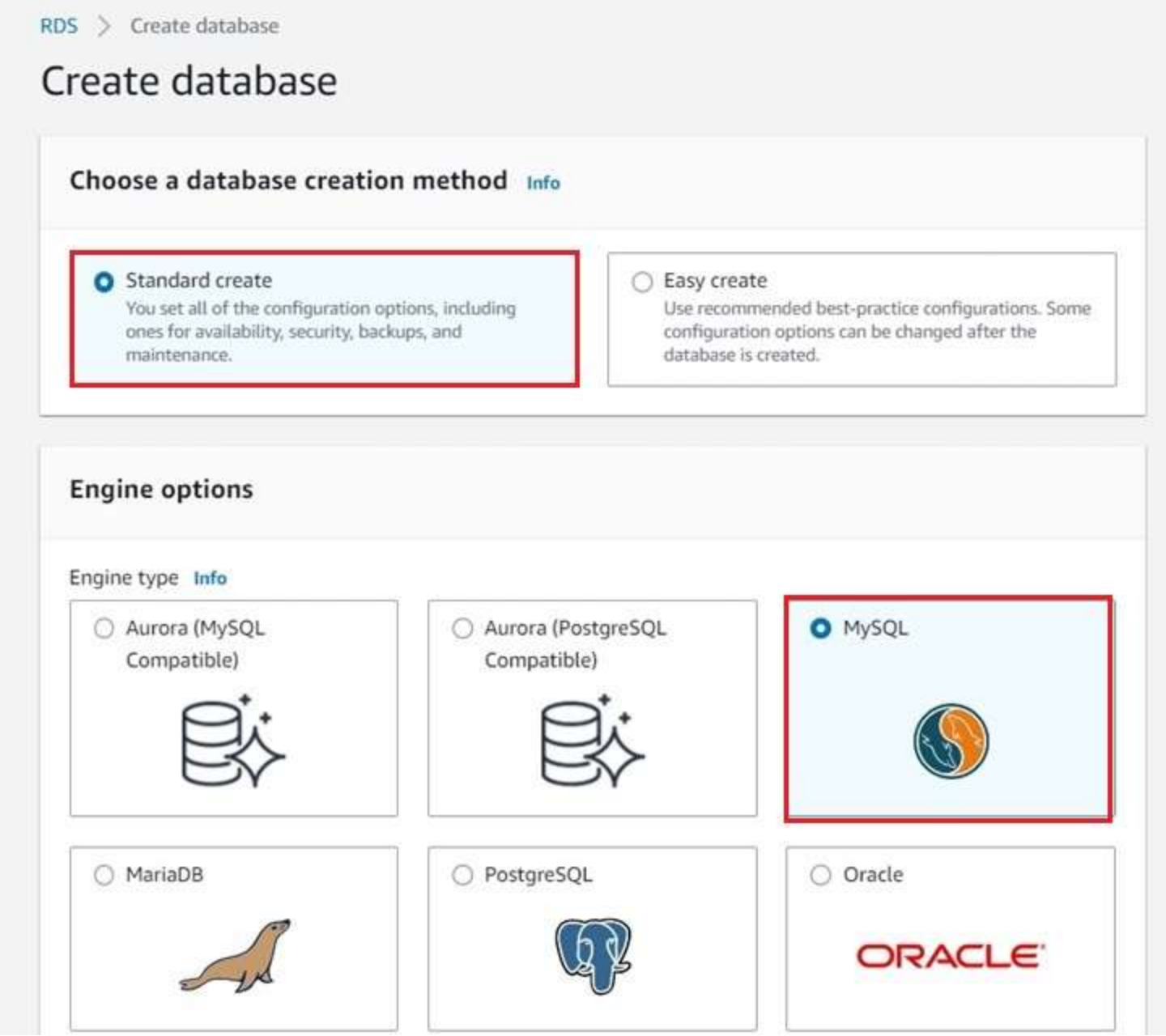
STEP 5: CREATE AND EDIT ROUTE TABLE

For the Database Tier, the route table **(RouteT_Private_3Tier_DB)** was already created and edited to route traffic from the **Database Tier private subnet (Private_Sub_DB_1b)** and from the **Application Tier private subnet (Private_Sub_App_1b)** to NAT gateway **(NATGW2_3_Tier_Architecture)** that resides in the Web Tier public subnet **(Public_Sub_Web_1b).**

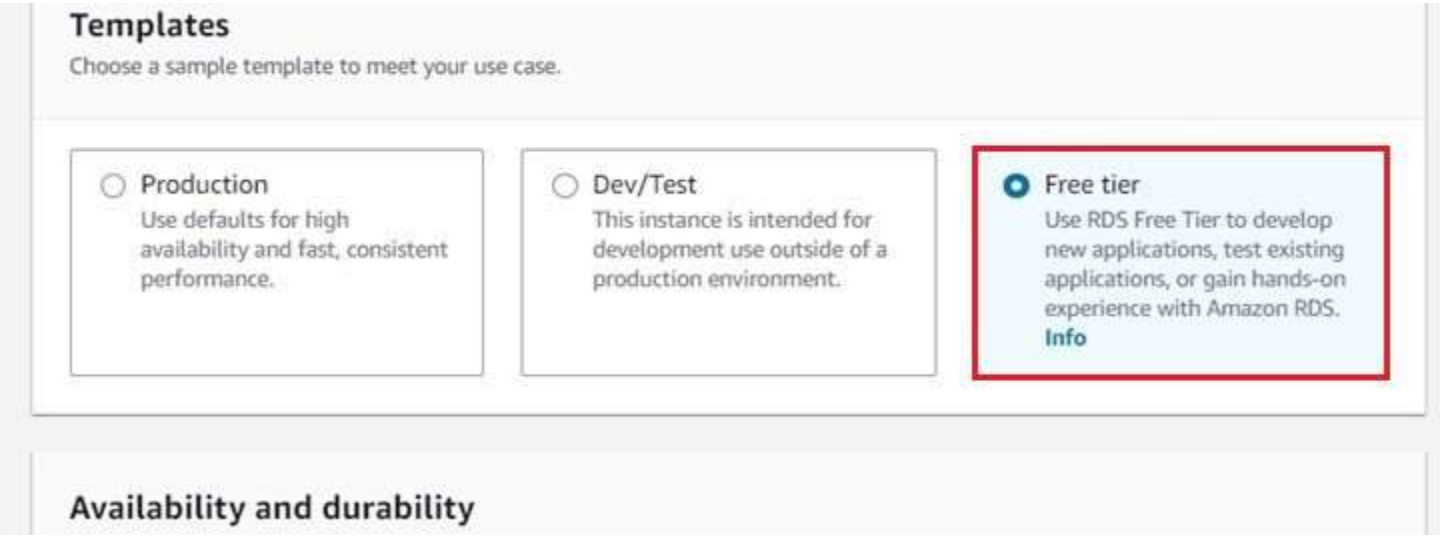
In other words, the Route Table named **(RouteT_Private_3Tier_DB)** consumes traffic that is routed to it from the Database Tier private subnet **(Private_Sub_DB_1b)** and from the Application Tier private subnet **(Private_Sub_App_1b)** and then wires these traffic to NAT gateway **(NATGW2_3_Tier_Architecture)** that resides in the Web Tier public subnet **(Public_Sub_Web_1b).** So, we currently do not need to create new route tables at this point.

STEP 6: CREATE DATABASE INSTANCE

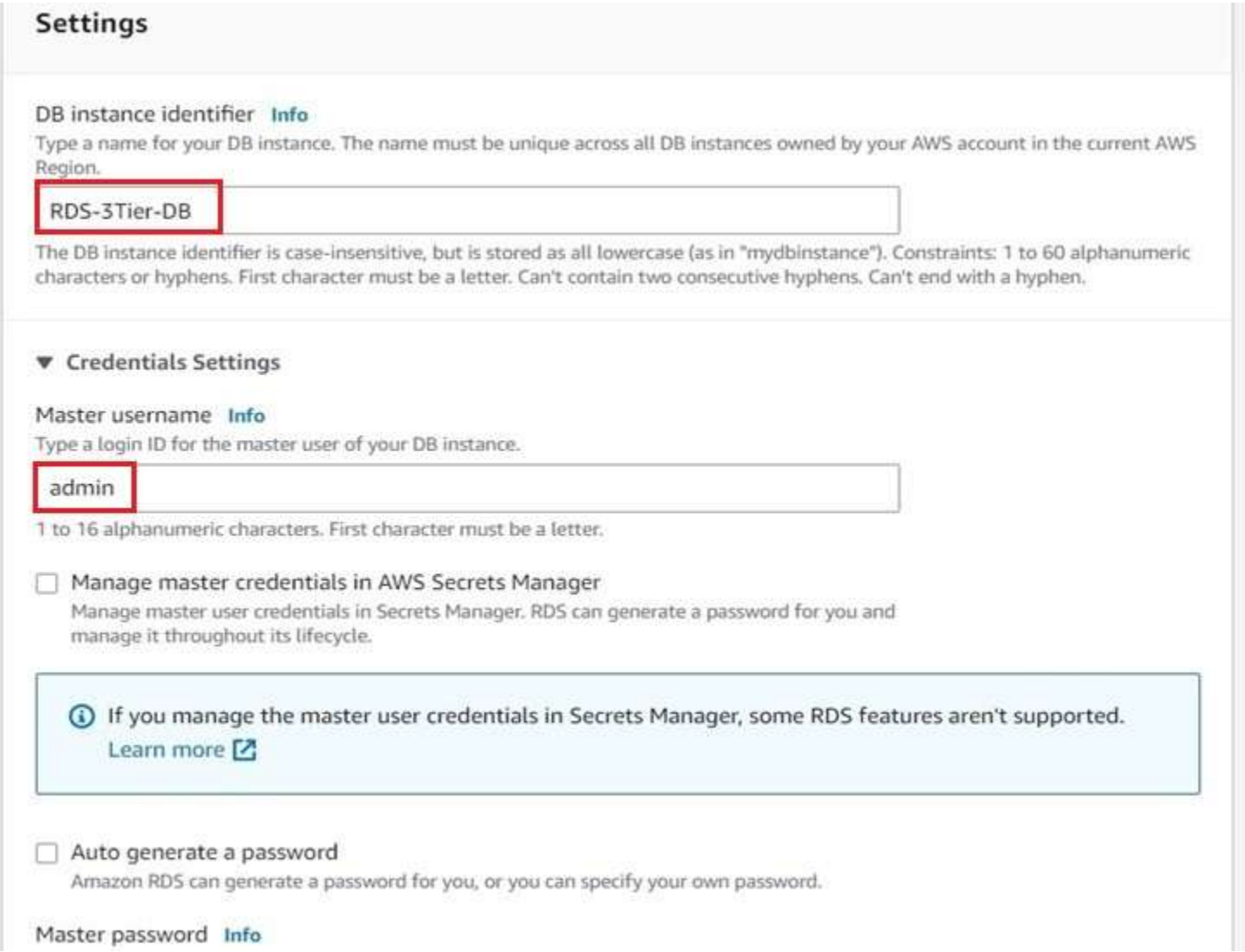
Search for Databases, navigate to RDS, click on create database, select "Standard create" and Choose MySQL.



Specify the Free tier template.

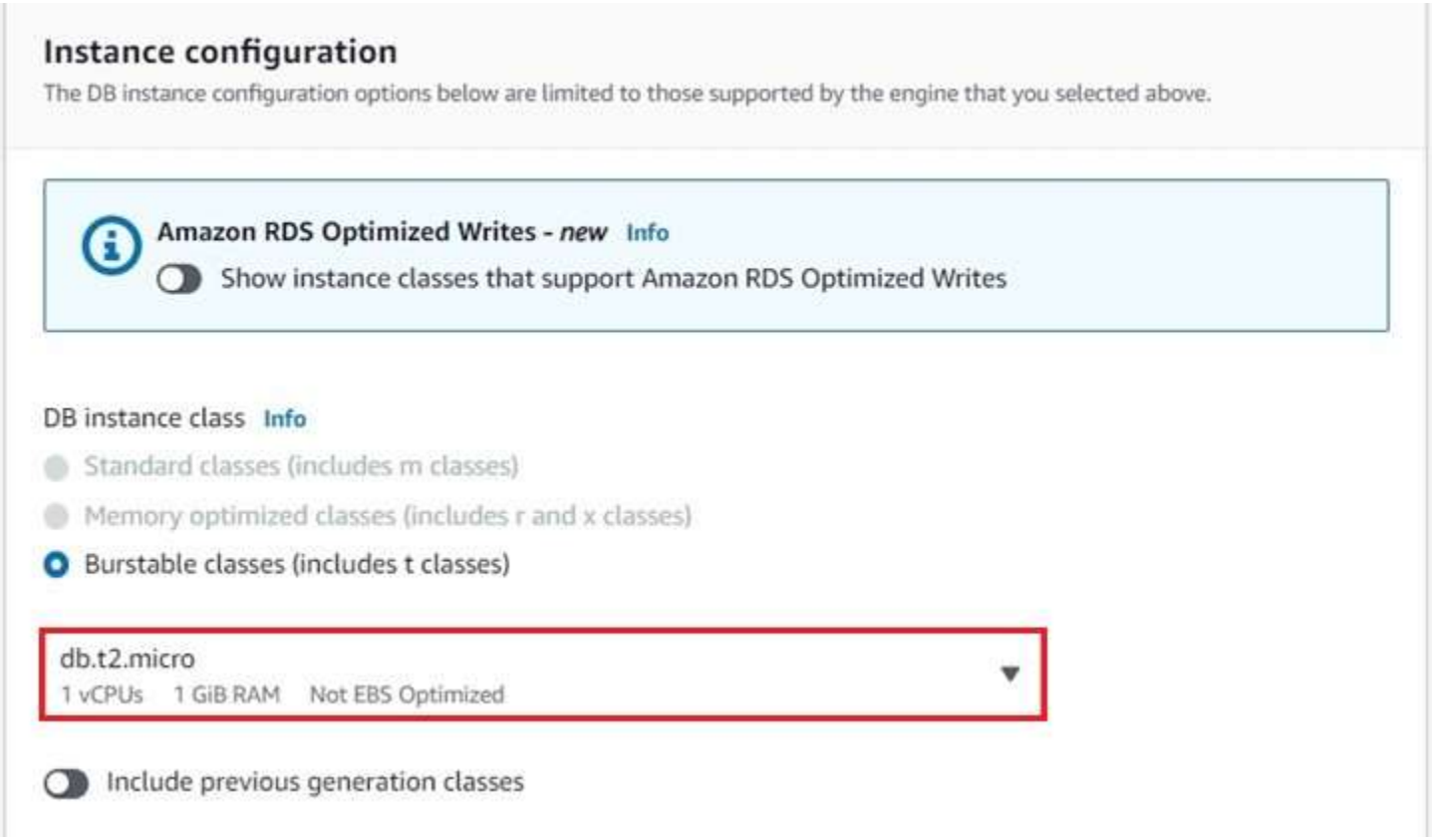


Then enter the name (It is important to note that First character must be a letter. Can't contain two consecutive hyphens. Can't end with hyphen.)



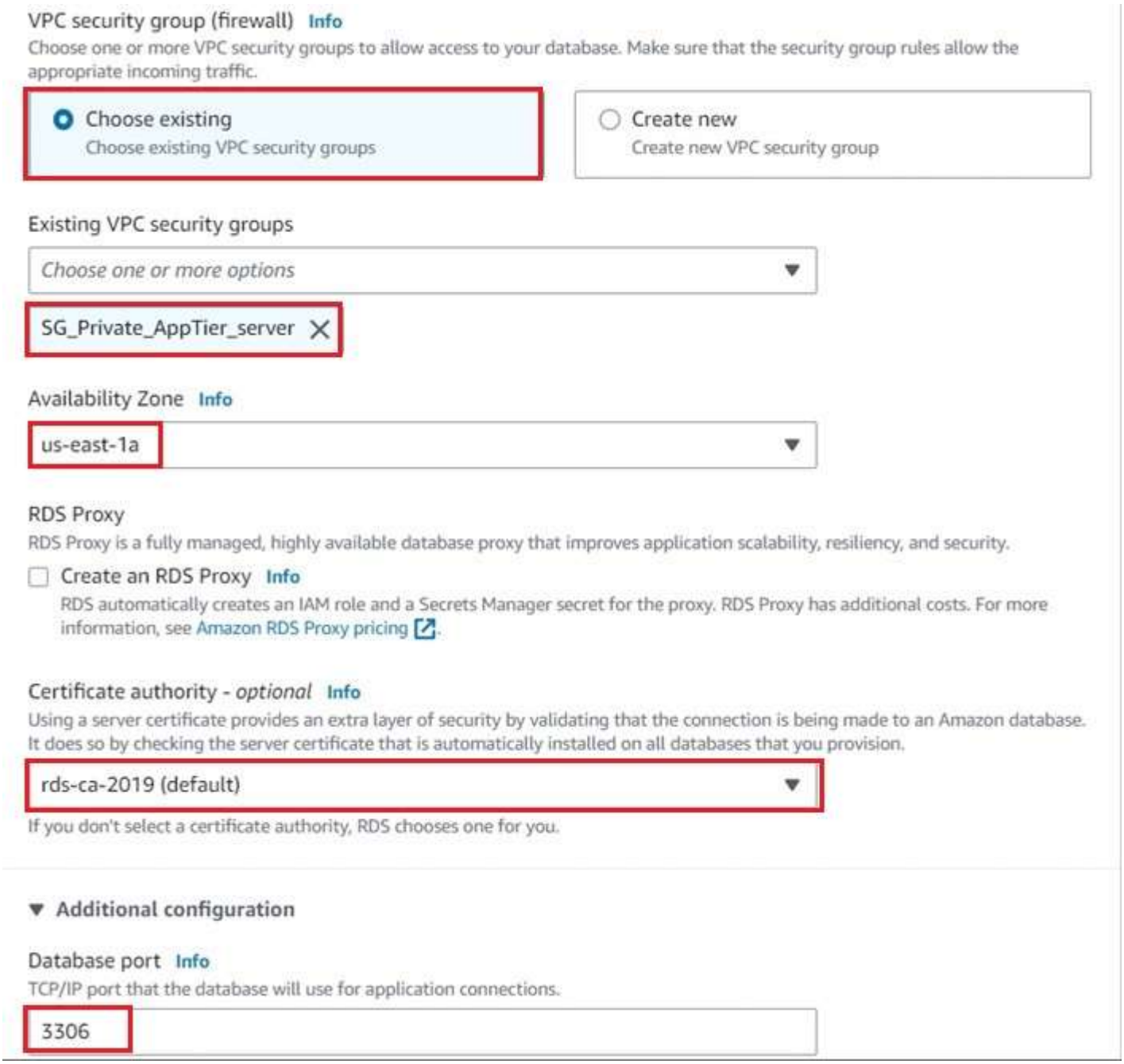
Next step will be to enter username and password.

Database Instance class: Burstable class (db.t2.micro)



STEP 7: SECURITY GROUP

For VPC security group field, let's use an already existing security group, that way we wouldn't have to create a new one. Therefore, let's use the **Application Tier security group (SG_private_AppTier_server)** would serve the purpose, and it should be referenced as the source security group.



Scroll down to Additional configuration, you can choose back up options. All we need to do here is give it a name and maintain the default values.

Connectivity Info

Compute resource

Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.

- (●) **Don't connect to an EC2 compute resource**
  Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.
- ( ) **Connect to an EC2 compute resource**
  Set up a connection to an EC2 compute resource for this database.

Virtual private cloud (VPC) Info

Choose the VPC. The VPC defines the virtual networking environment for this DB instance.

VPC_3_Tier_Architecture (vpc-08328352666abbbb8)
6 Subnets, 2 Availability Zones

Only VPCs with a corresponding DB subnet group are listed.

ⓘ After a database is created, you can't change its VPC.

DB subnet group Info

Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.

Create new DB Subnet Group

Public access Info

- ( ) **Yes**
  RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.
- (●) **No**
  RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

▼ Additional configuration

Database options, backup turned on, backtrack turned off, maintenance, CloudWatch Logs, delete protection turned off.

Database options

Initial database name **Info**

RDS_3Tier_DB_Private

If you do not specify a database name, Amazon RDS does not create a database.

DB parameter group **Info**

default.mysql8.0

Option group **Info**

default:mysql-8-0

Backup

☑ Enable automated backups

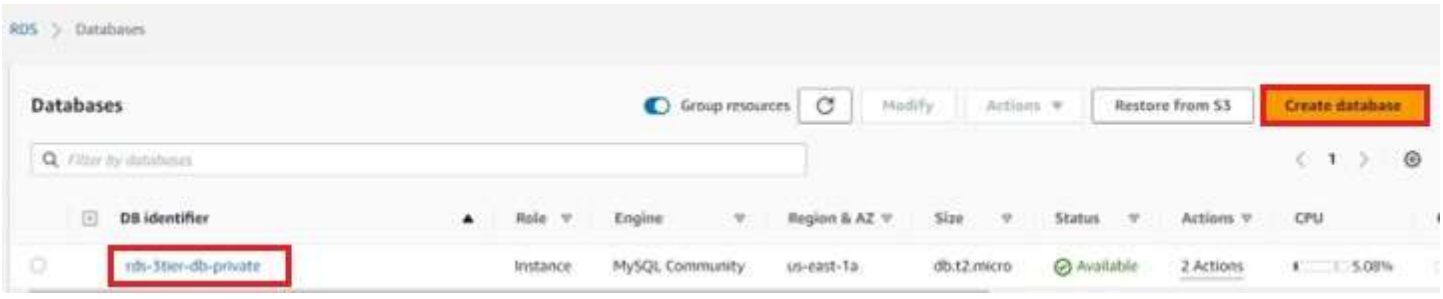
Creates a point-in-time snapshot of your database

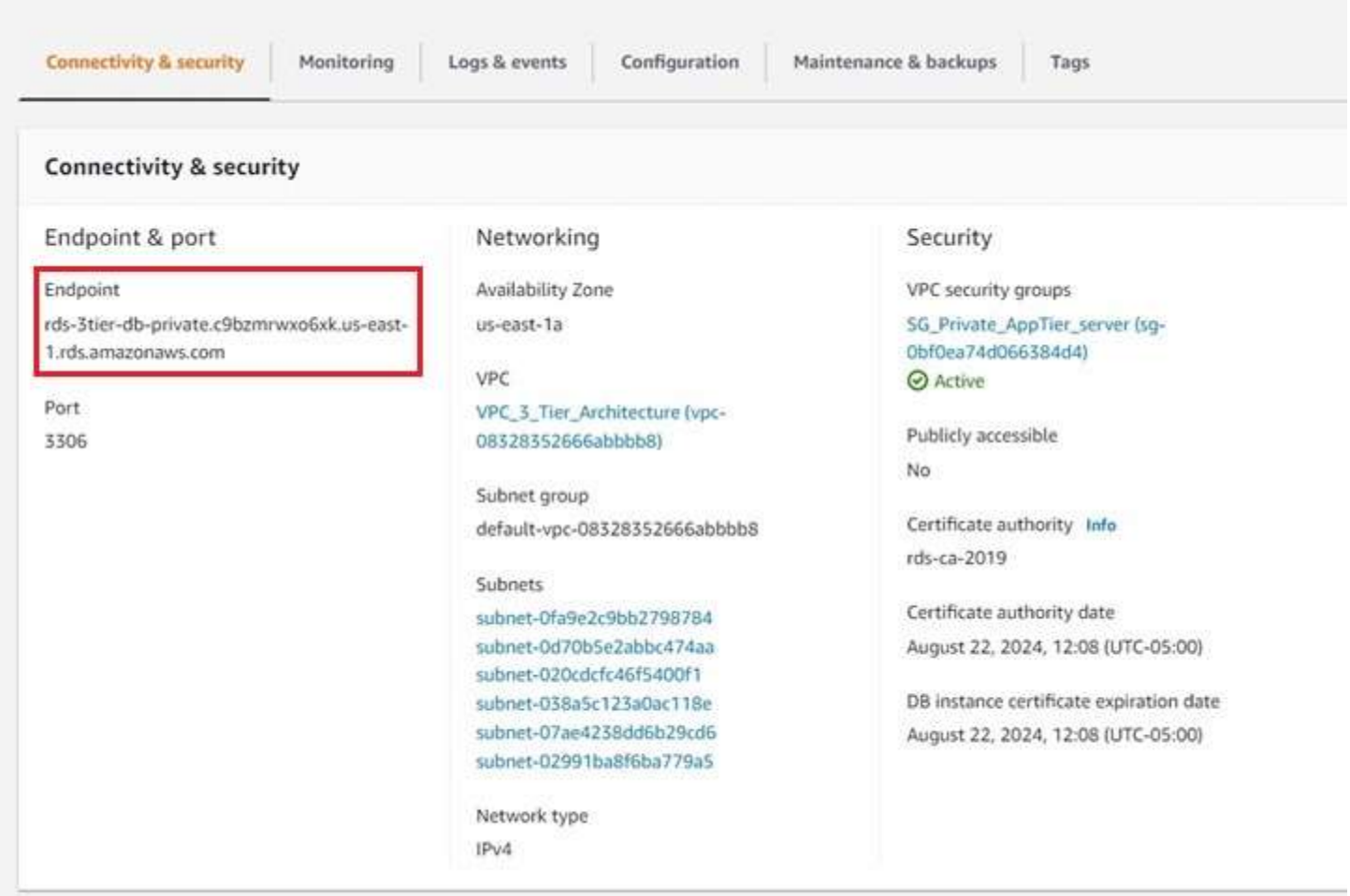
⚠ Please note that automated backups are currently supported for InnoDB storage engine only. If you are using MyISAM, refer to details here.

Backup retention period **Info**

The number of days (1-35) for which automatic backups are kept.

7 days





Database successfully created.

Finally, we are done with the project, and everything is working perfectly as supposed. We can ping the App Servers and Database Servers from the Web Tier Servers.

If you've followed along with me until the end of this project, please delete all resources that were deployed in building this project as some of the used AWS services will incur charges on your account if left running. Delete or detach resources such as elastic IP, EC2 instances, databases, NAT gateways etc. where applicable.

The key objectives of the AWS 3-tier architecture is to create a robust 3-tier architecture that is scalable, secure, highly available, reliable, and flexible for deploying web applications on the AWS cloud, ensuring efficient resource utilization and optimal performance. This architecture divides the application infrastructure into three distinct tiers or layers: the presentation layer, the application layer, and the data layer.

Presentation Tier: Also known as the user interface tier, this layer focuses on the presentation of the application to users. It typically consists of web servers or client-side applications that handle user interactions and display the application's user interface.

Application Tier: This layer contains the business logic and application processing components. It handles the core functionality of the application, including processing user requests, executing business rules, and interacting with databases or other external systems. It can be implemented using various services such as AWS Lambda, AWS Elastic Beanstalk, or EC2 instances running application servers.

Data Tier: The data tier is responsible for managing and storing the application's data. It typically involves databases or data storage services like Amazon RDS (Relational Database Service), Amazon DynamoDB (NoSQL database), or Amazon S3 (Simple Storage Service). This tier ensures data persistence and provides efficient data retrieval and storage capabilities.

The key benefits of an AWS 3-tier architecture are:

Scalability: The architecture allows for independent scaling of each tier. This means that you can scale each layer horizontally or vertically based on the specific needs of your application. This enables efficient resource allocation and ensures that the application can handle increased user demand or data volume.

High Availability: By distributing the application across multiple tiers and using AWS services such as Elastic Load Balancers and Auto Scaling, the architecture aims to achieve high availability. If one component fails, other components can continue to function, ensuring minimal downtime and uninterrupted service for users.

Fault Isolation: By separating different components into distinct tiers, the architecture provides fault isolation. If one component experiences issues or failures, it is less likely to affect other parts of the application. This enhances the overall reliability and resilience of the system.

Security: The architecture allows for the implementation of security measures at each tier. This includes using AWS security services like Identity and Access Management (IAM), Virtual Private Cloud (VPC), and Network Access Control Lists (ACLs) to enforce security policies and control access to resources.

Most Importantly, I want to say thank you for taking the time to follow and read through my project.

I hope you liked it and please don't hesitate to reach out if you have any questions, comments, or suggestions!

See you soon!