Faculty of Mathematical, Physical and Natural Sciences of Tunis

Faculté des Sciences Mathématiques, Physiques et Naturelles de Tunis

كلية العلوم للرياضيات والفيزياء والطبيعيات بتونس

AWS Project

Realized by : Oussama Maaroufi

Year 2022/2023

IF5

Plan :

I. Introduction

II. Steps :

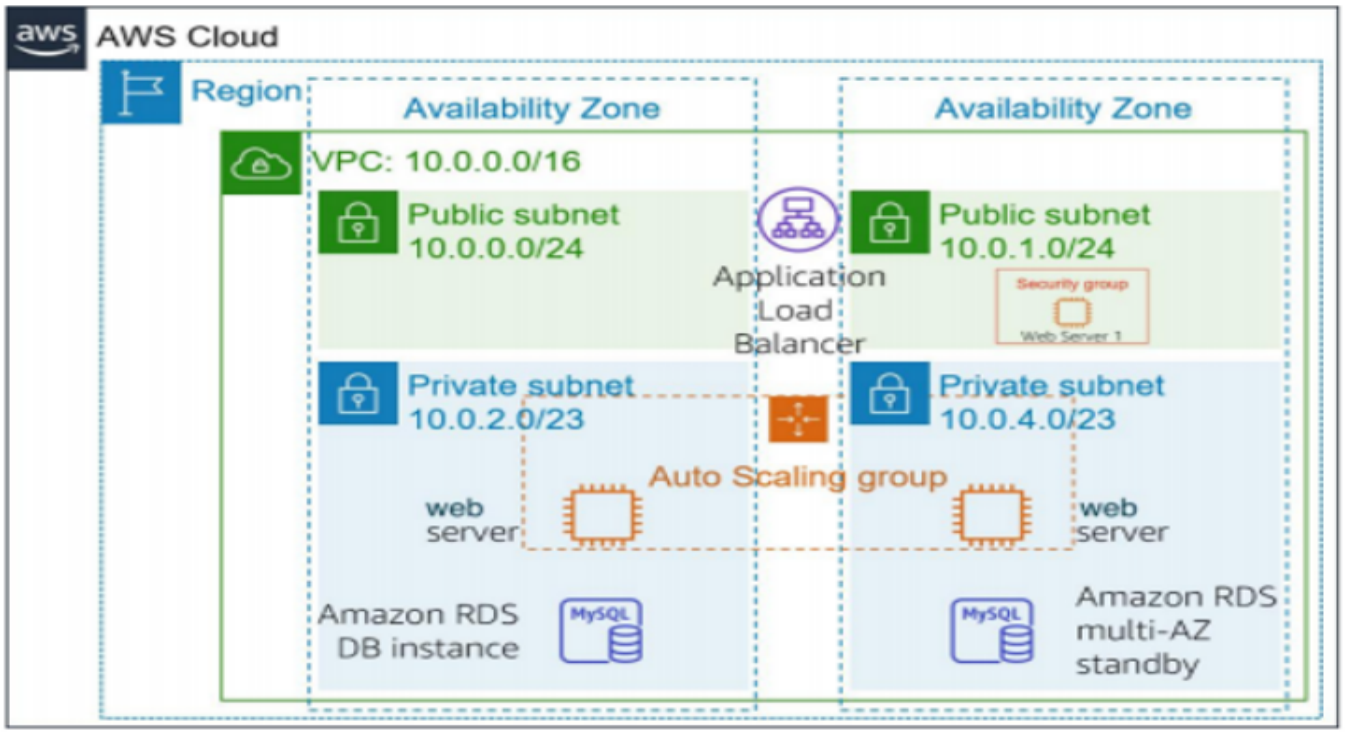
i. Create the VPC for our project
ii. Create Public/Private Subnets:
iii. Overview about Route Table
iv. Create and Attach Internet Gateway
v. Create a Route Table for the public subnets
vi. Create NAT Gateways
vii. Create Security Groups
viii. Create a LoadBalancer
ix. Create an autoscaling group
x. Create Database Instance RDS - MYSQL
xi. Deploy A React App In EC2 Instance
xii. Create Machine Bastion

III. Conclusion

I. Introduction

High availability (HA) is the elimination of single points of failure to enable applications to continue to operate even if one of the IT components it depends on, such as a server, fails . So in this project we decided to create a height-available architecture using AWS services such as EC2, RDS , Auto Scaling Group Application Load Balancer ,Vpc etc .

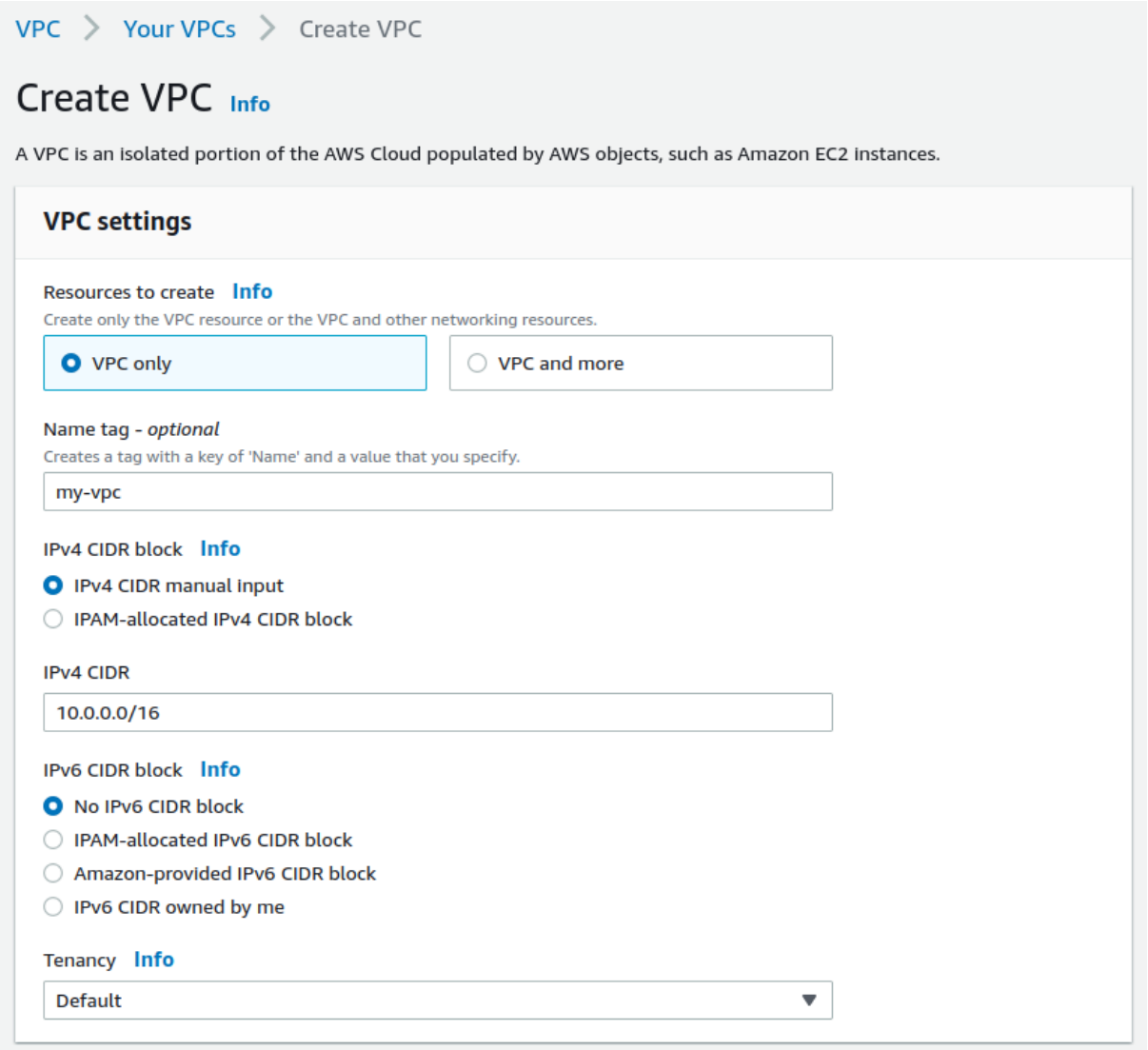
The Architecture that we want to build is shown below

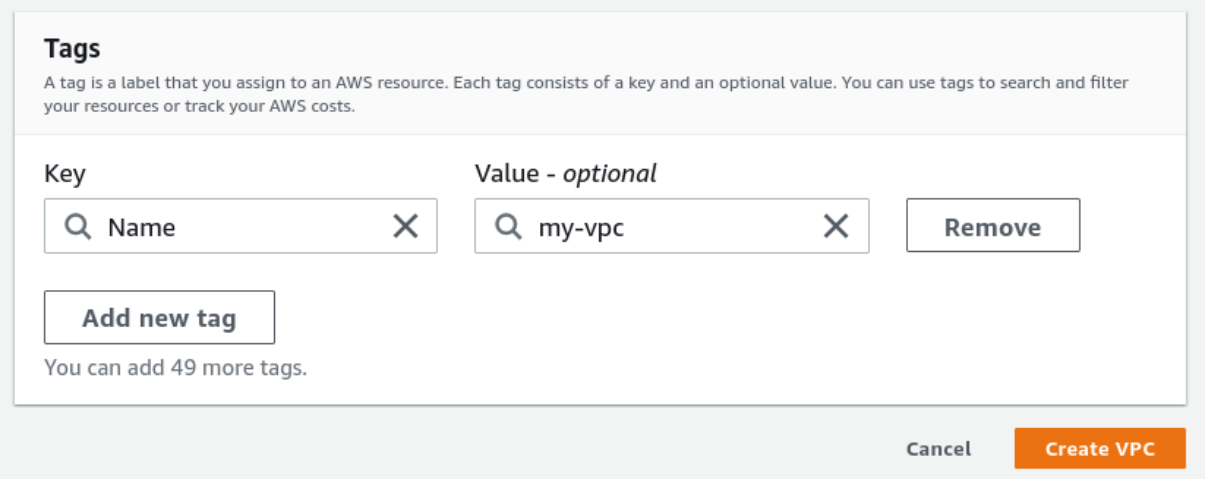


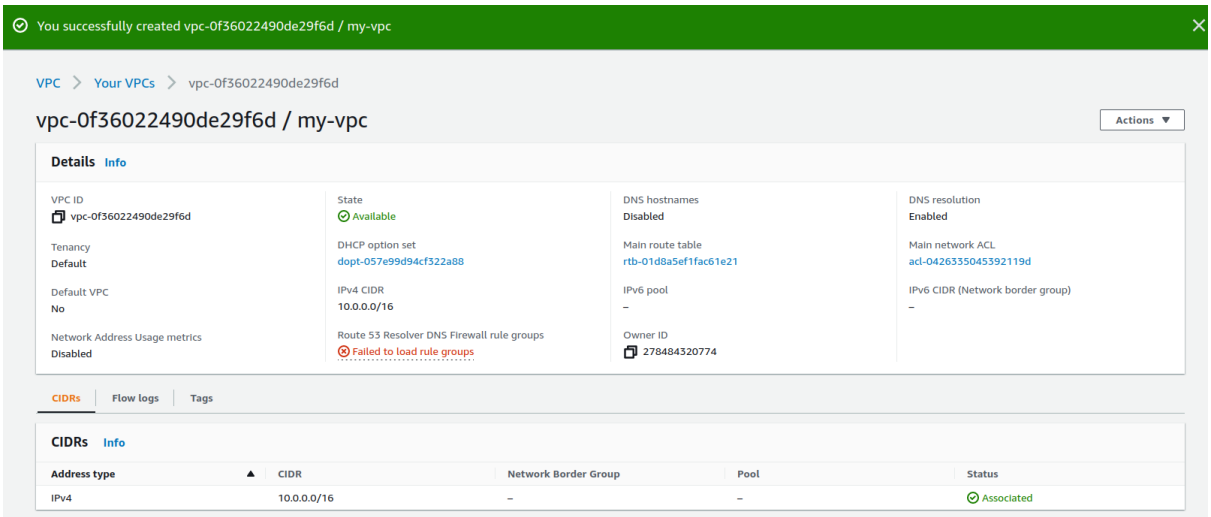
II. Steps

Step 1: Create the VPC for our project

Firstly from the management console,we verify the region that we are currently in and where we want to launch this VPC. We are currently in the ***us-east-1*** northern Virginia region.
Navigate to VPC in the AWS console. Click “Create VPC.” Choose “VPC only” and we enter an optional name tag. Name tags are useful to help keep our projects organized. We choose the IPv4 CIDR block. I have chosen to manually enter a CIDR block of 10.0.0.0/16. Adjust tenancy and tags as desired, then click “Create VPC.”



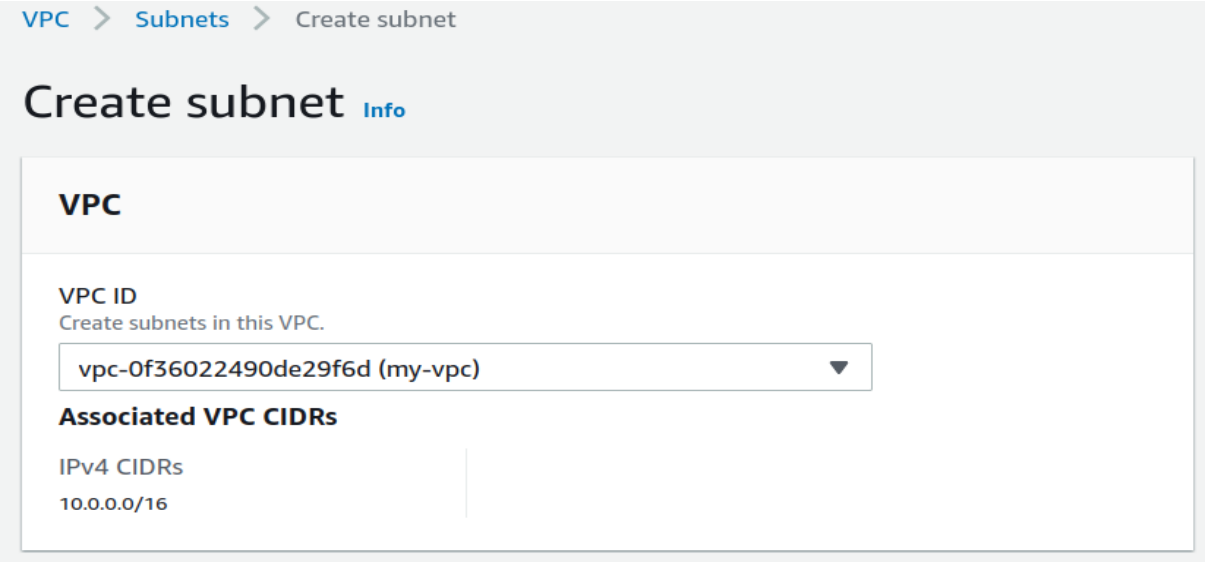




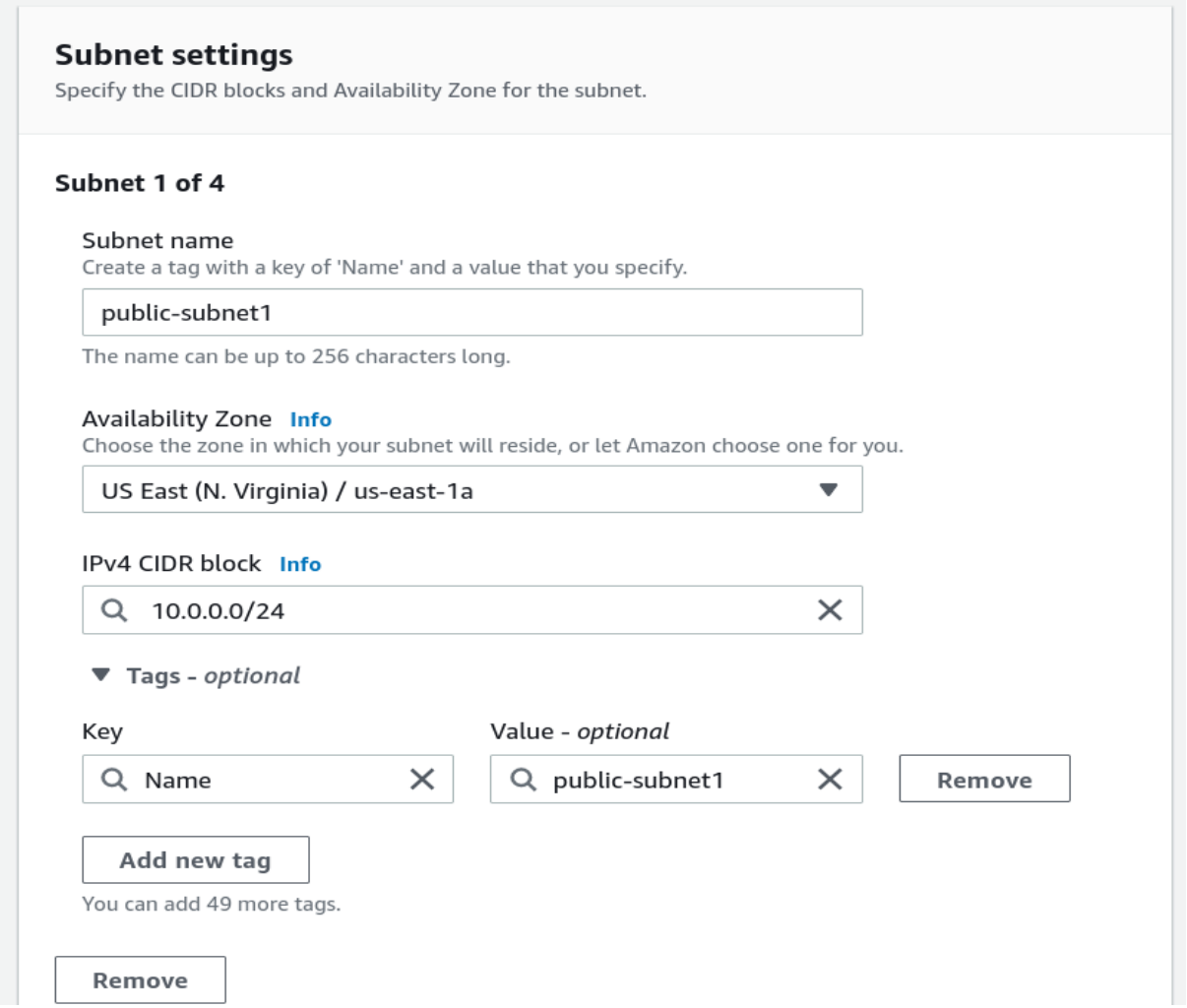
Step 2: Create Public/Private Subnets

Now let's create two public subnets with the following CIDR blocks: **10.0.0.0/24** , **10.0.1.0/24**, These will be created within two different availability zones and two private subnets with following CIDR Blocks: **10.0.2.0/23 , 10.0.4.0/23** .

By creating multiple subnets across multiple availability zones, we increase **reliability** and **availability**. In the VPC console, on the left side menu, select "Subnets," then click the orange "Create subnet" button. We want to create subnets within the VPC that We created in Step 1, so we choose that VPC from the drop down menu.



We are able to create all four subnets at the same time, and they will be created within the VPC above. Enter the names of the new subnets and their IPv4 CIDR block information. Choose a different availability zone for each subnet.



Subnet 2 of 4

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

public-subnet2

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

10.0.1.0/24

Tags - *optional*

Key	Value - *optional*	
Name	public-subnet2	Remove

Add new tag

You can add 49 more tags.

Remove

Subnet 3 of 4

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

private-subnet1

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

10.0.2.0/23

Tags - *optional*

Key	Value - *optional*	
Name	private-subnet1	Remove

Add new tag

You can add 49 more tags.

Remove

Subnet 4 of 4

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

private-subnet2

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

10.0.4.0/23

Tags - *optional*

Key	Value - *optional*	
Name	private-subnet2	Remove

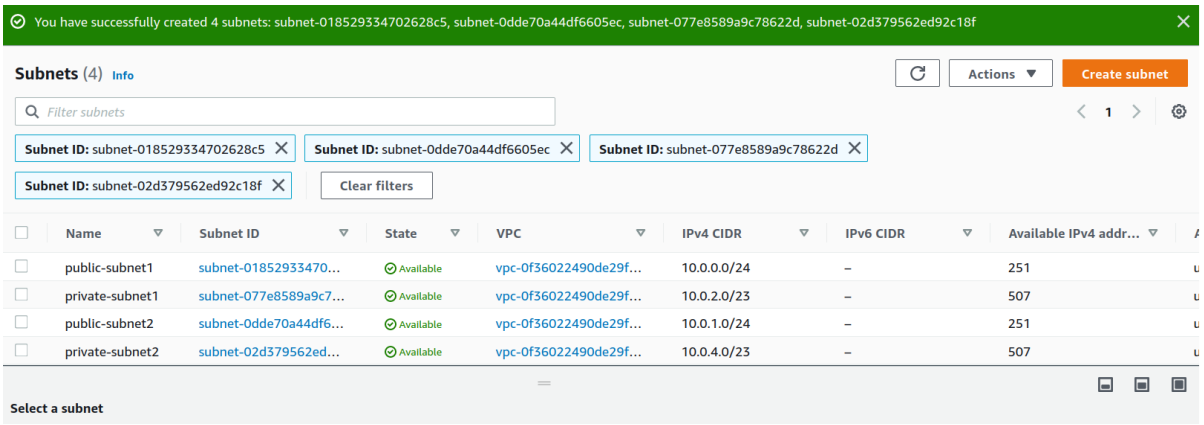
Add new tag

You can add 49 more tags.

Remove

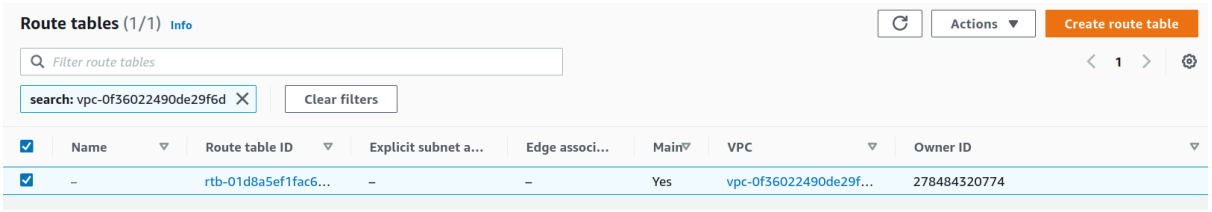
Add new subnet

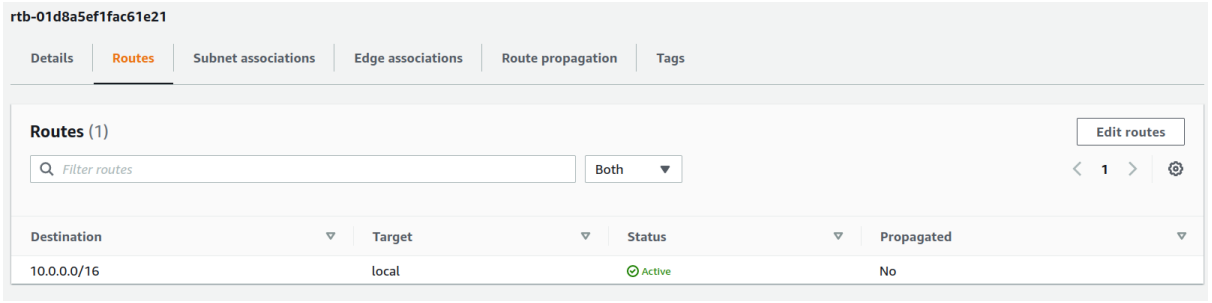
Cancel | Create subnet



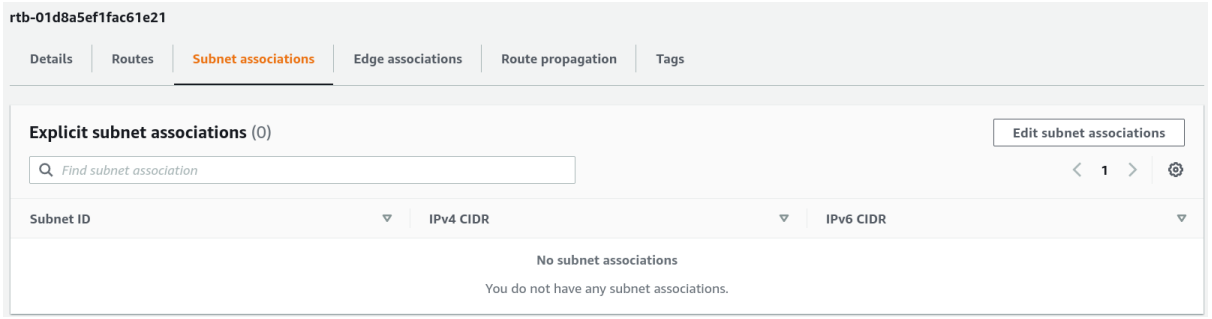
Step 3: Overview about Route Table

When we create a VPC it automatically comes with a main (or default) route table that is responsible for directing network traffic. We can create a custom route table if desired, and associate subnets We created to the custom route table. If we do not manually associate the subnets to a custom route table, they will be automatically associated to the main route table. Routes in the table have both a destination and a target.
The target is typically the internet gateway that is associated with our VPC for public subnets and local for private subnets . For the example above, if We select “Route tables” from the left side menu in the VPC console, We will see my default route table with the VPC We created previously.
The default Route Table :





If We click on “Subnet associations,” We can see that the four subnets we created within the VPC are automatically associated with this default route table, which has a local target.

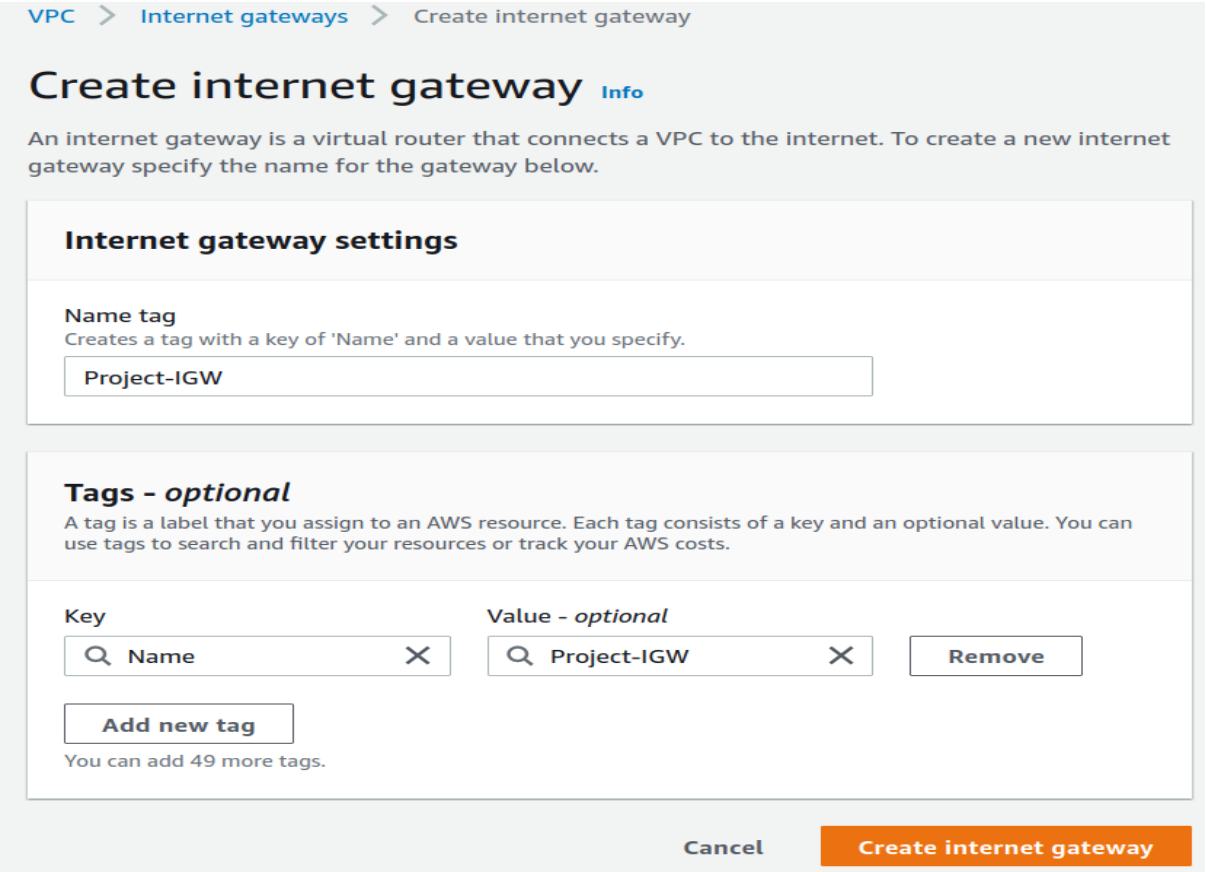


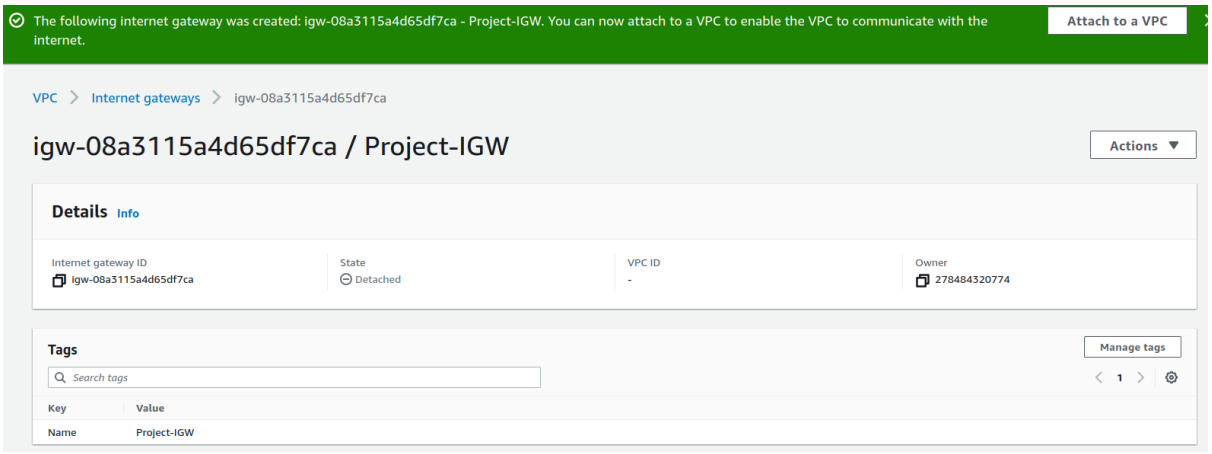
As we see above the default route table only contains one route with target locale and that means that there is no internet traffic available which indicates that our subnets are private by default .

So what we need to do is create another route table with a route that has an internet gateway as its target and associate public subnets to it and keep private subnets associated to main route table ., But first let's Create an Internet gateway and attach it to our VPC .

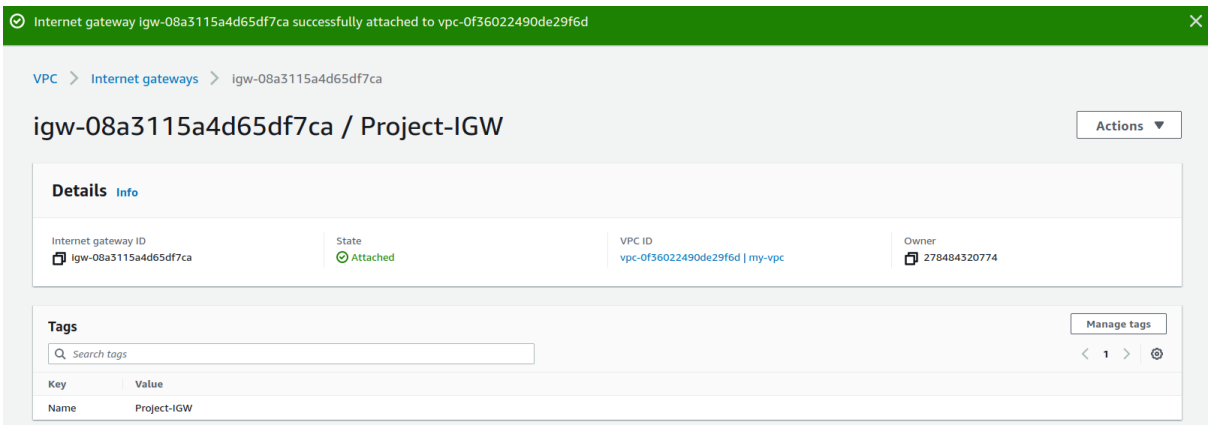
Step 4 : Create and Attach Internet Gateway

In order to allow resources in your VPC to communicate with the internet, you must create and attach an internet gateway. To do this, navigate to "Internet gateways" on the left side menu in the VPC console. Then select "Create internet gateway."

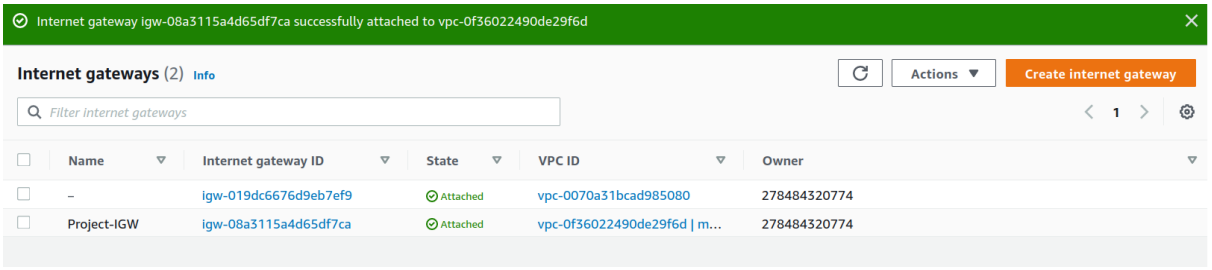




After We created an internet gateway, We need to attach it to the VPC we want to use. We will attach it to the VPC we created earlier. Click the "Attach to a VPC" button, then use the down arrow to select the VPC we created earlier. Click "Attach internet gateway."



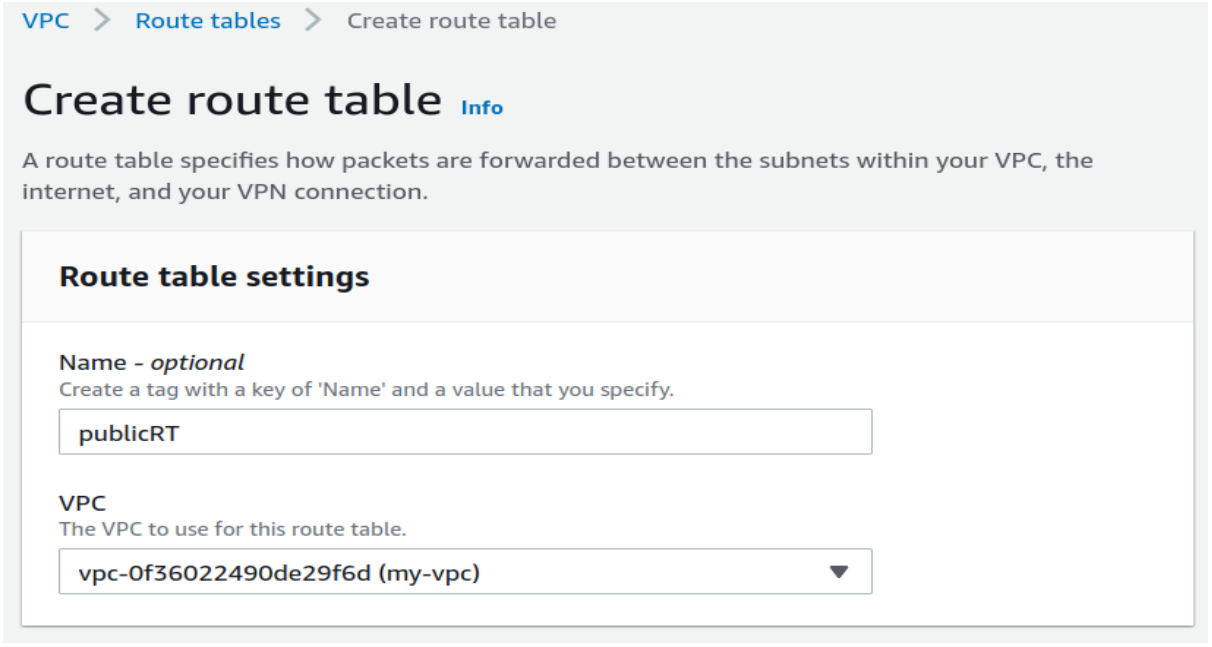
After attaching the internet gateway to our VPC, I should be able to see it listed under “Internet Gateways”.

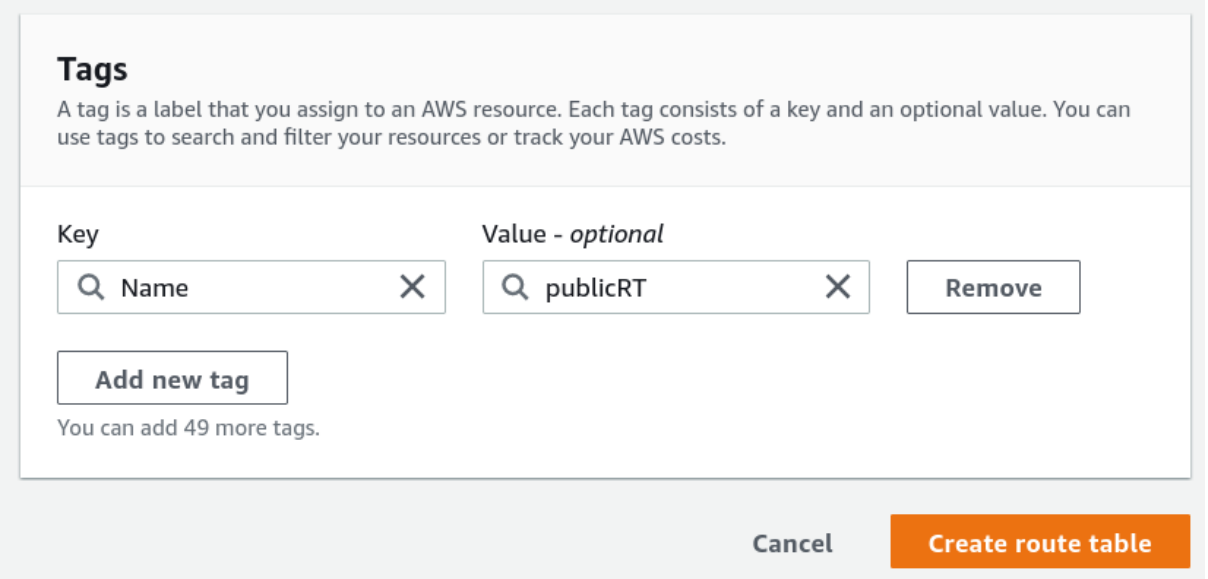


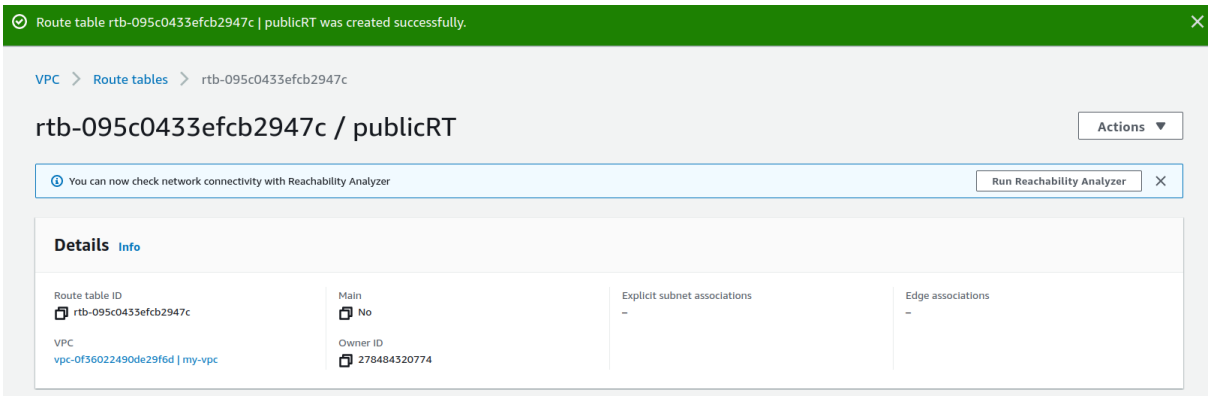
and now we are ready to create the Route Table

Step 5 : Create a Route Table for the public subnets

On the VPC console menu select *route table* and *create route table*. Name the route table and select our custom VPC to place it in.



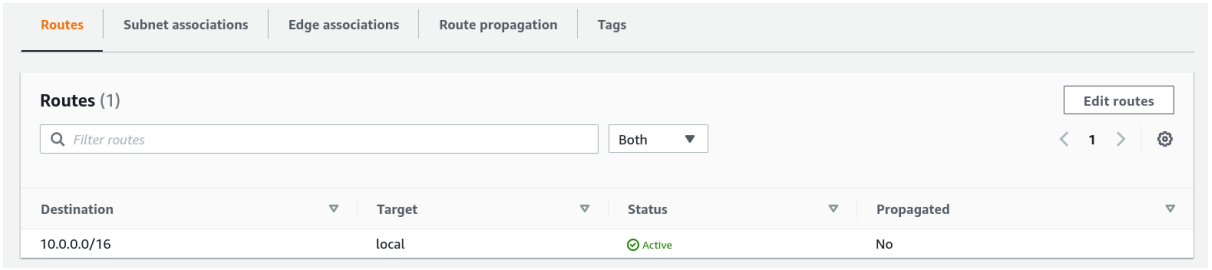


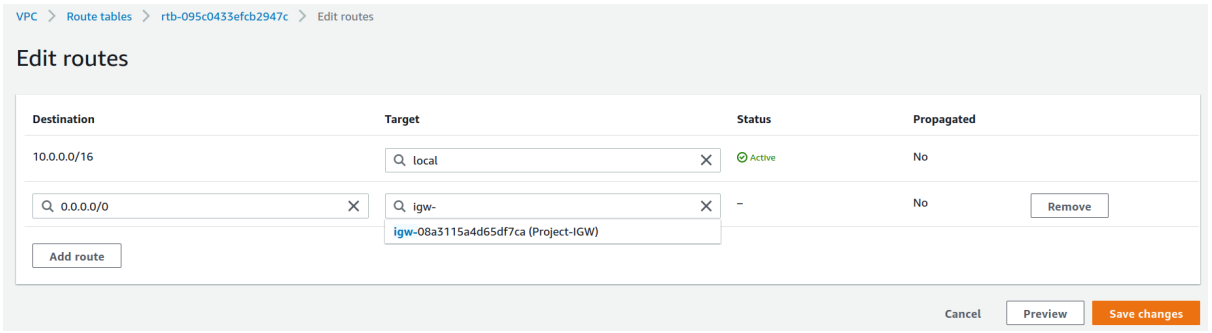


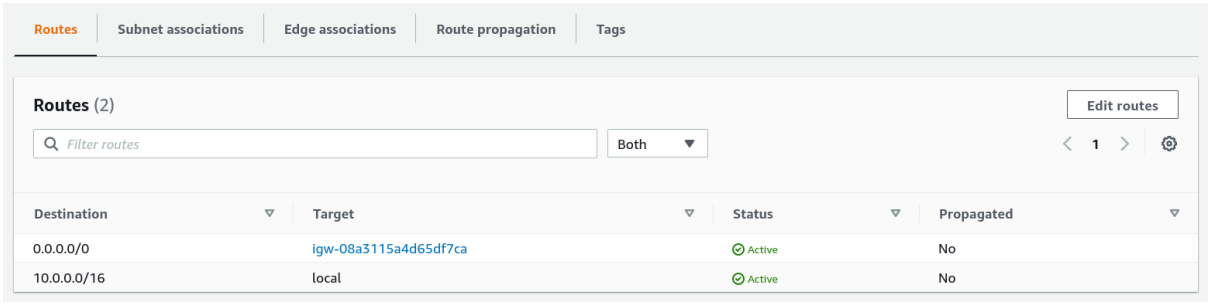
Now that we have created our public route table we need to create a **Public Route** for it. Select our Public route table and select the *routes tab*, then select *edit routes*.

Select *add route* For the destination select **0.0.0.0/0**. The destination of our public route will be to the public internet.

Under *target* select Internet gateway and select our custom **internet gateway** that we created earlier, save these changes.

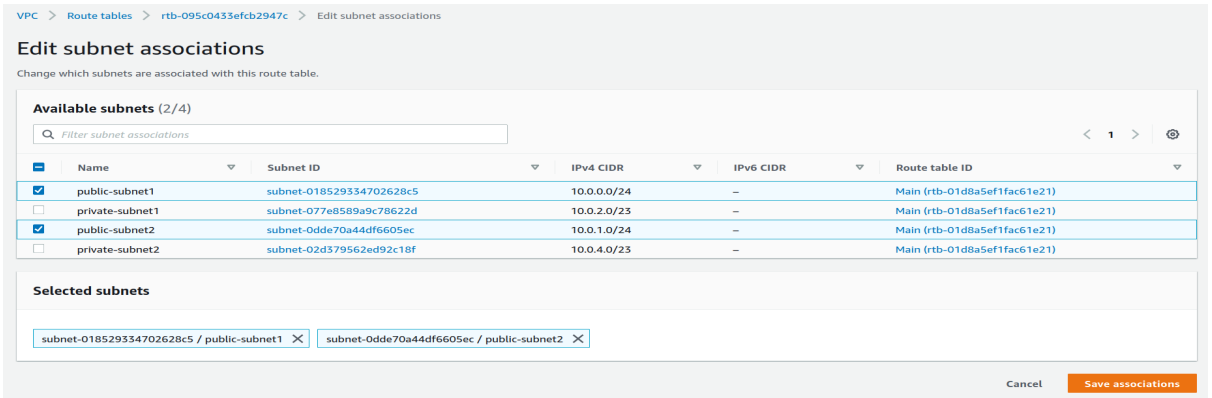


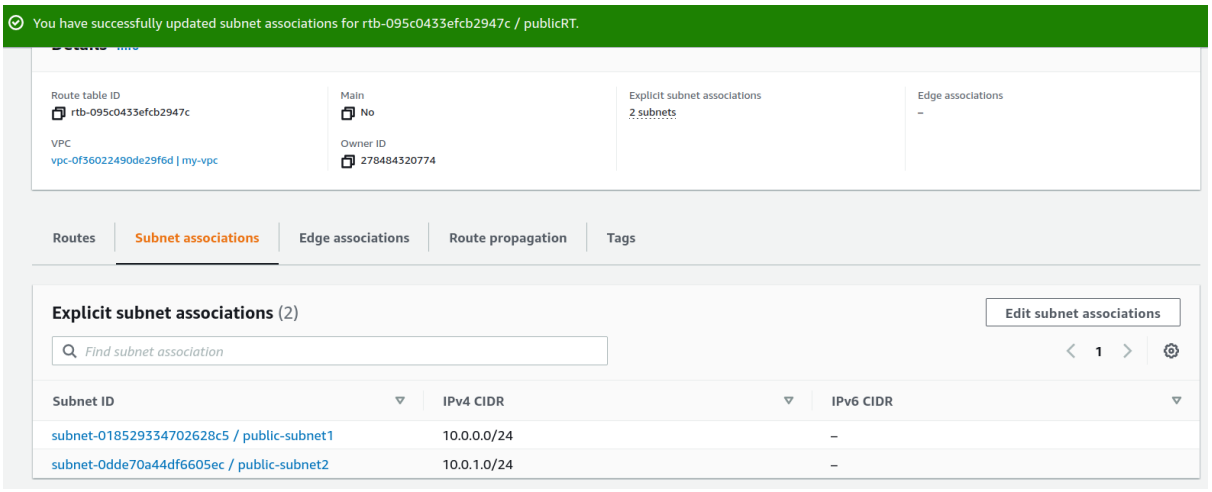




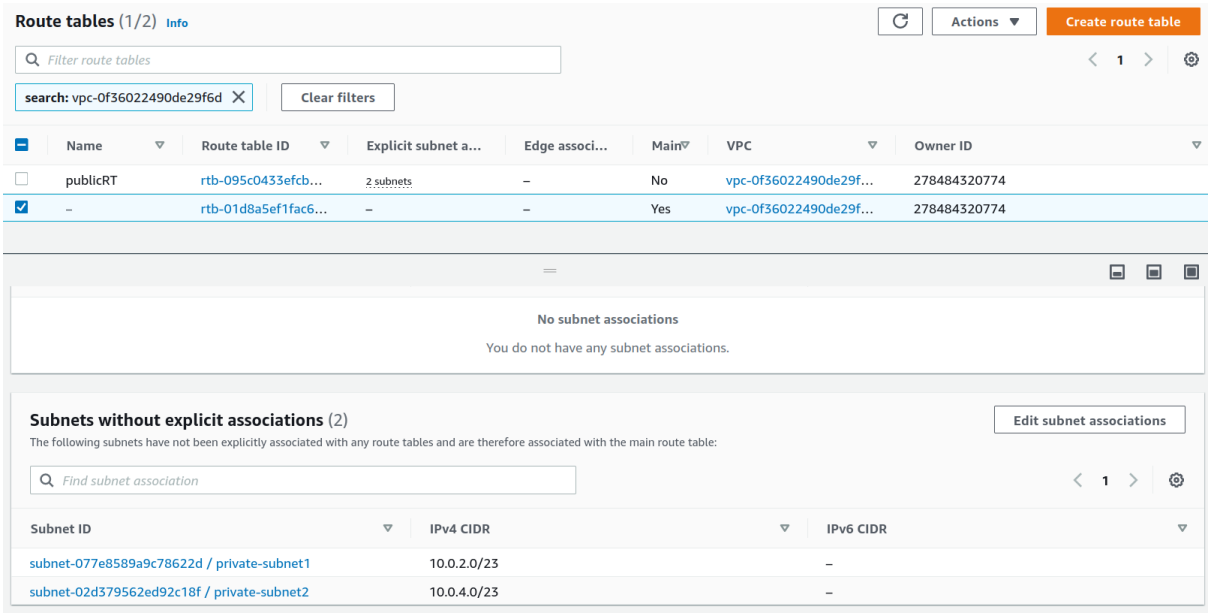
Now we will associate our route table with our public subnets.

Select *subnet associations* and then *edit subnet associations*. Select two public subnets and *save associations*.



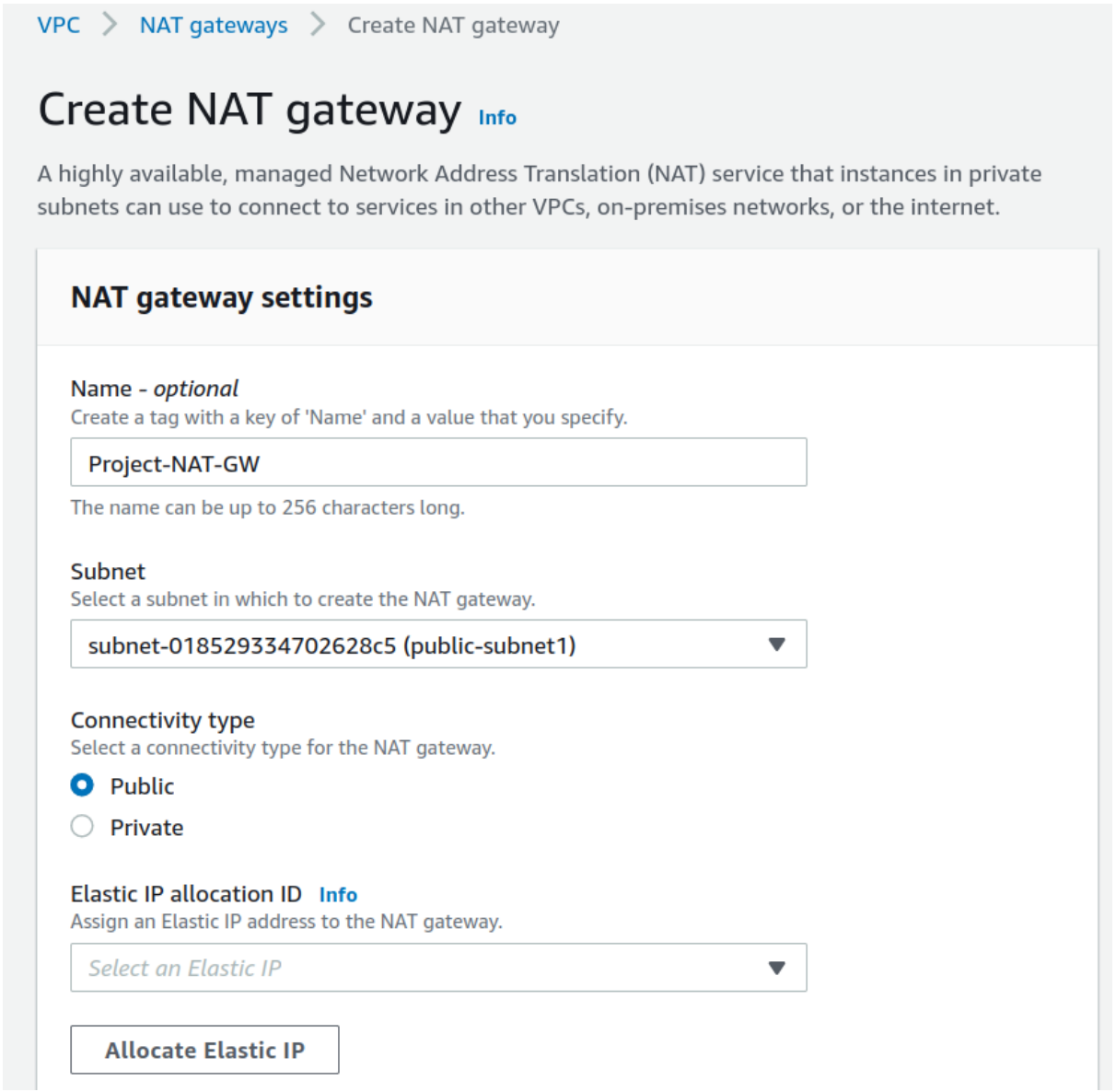


so now we have associated our public subnets to newly created Route Table and for private subnets they remain associated with the main Route Table as it's displayed below .

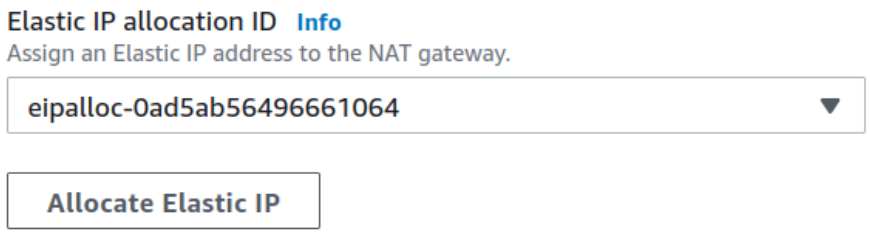


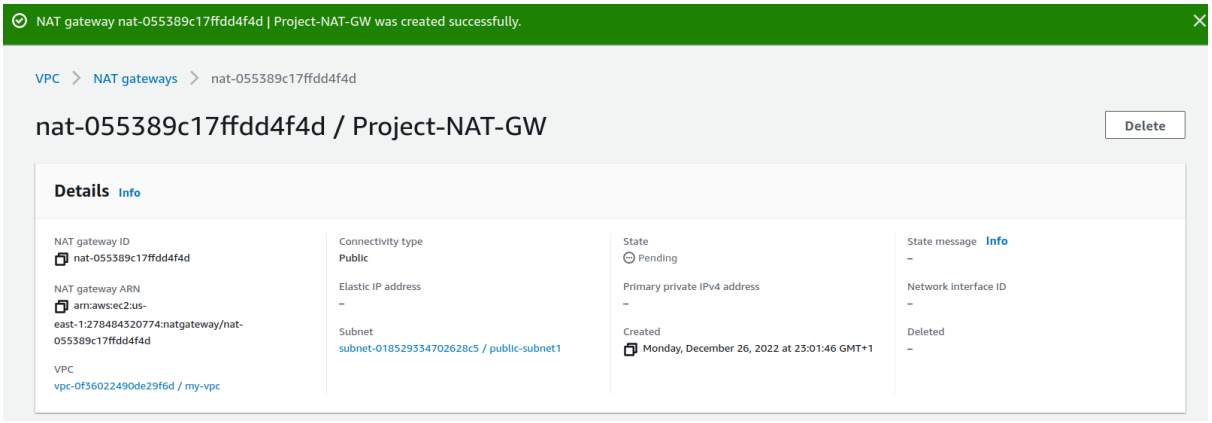
Step 6 : Create NAT Gateways

we need to create Nat Gateway and modify Main Route Table to allow patching and updating to our Ec2 instances that are launched in the private subnets



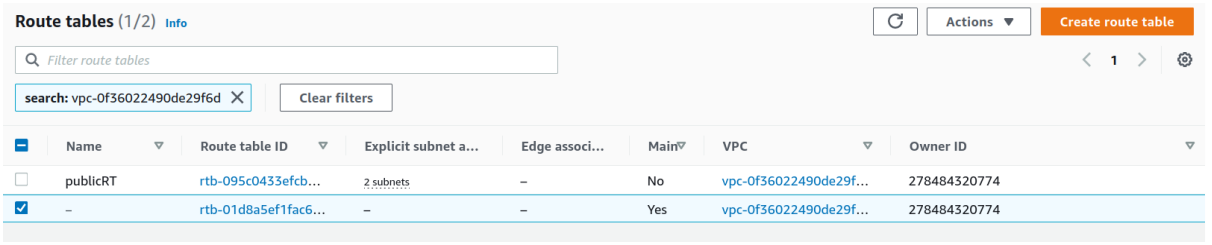
As we see we Need to Allocate an Elastic Ip for our Nat Gateway



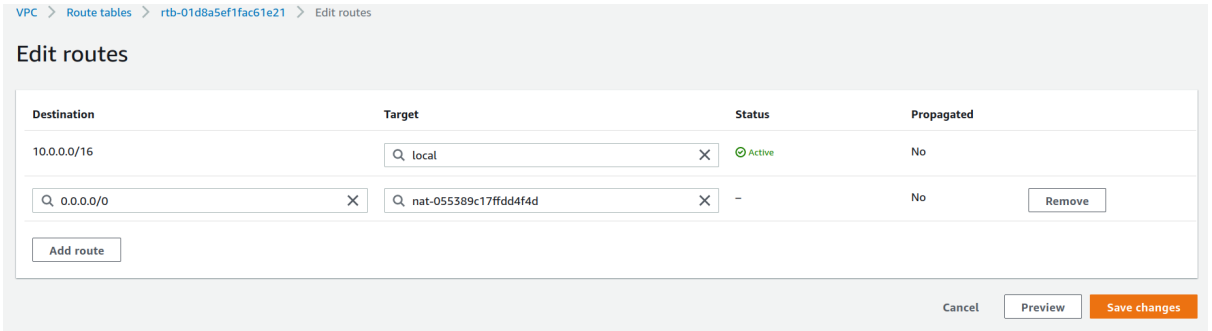


Now we Need to update the main route table which has private subnets associated with it by adding a new route with target nat gateway when the destination is the internet .

Choose the main route table and select edit routes

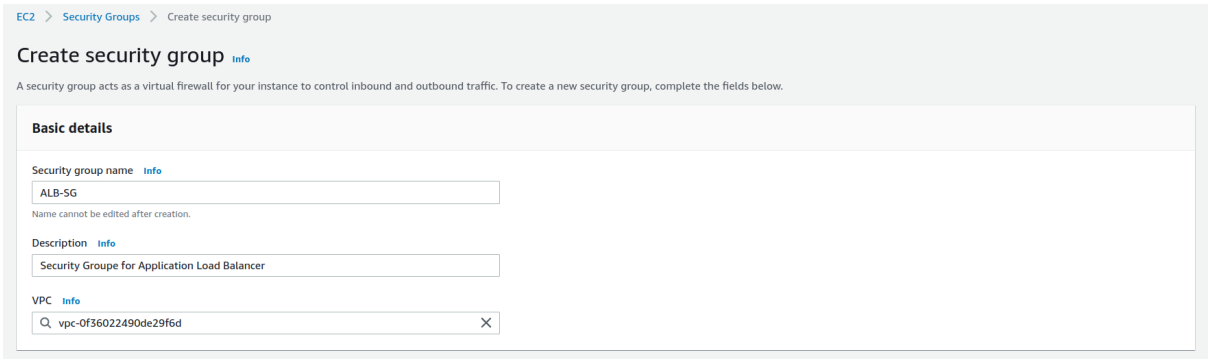


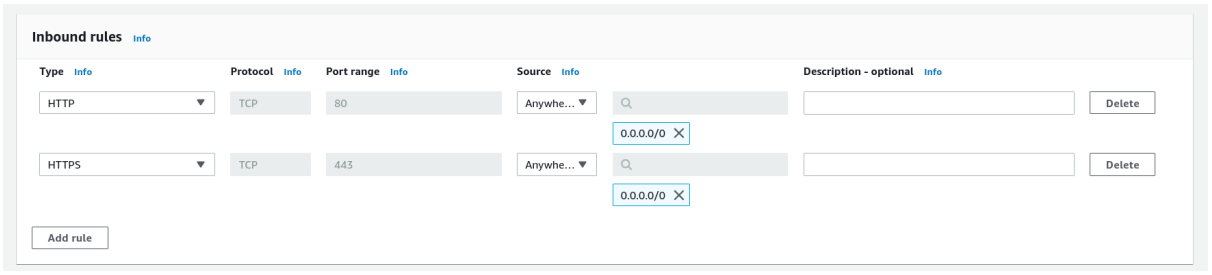
add a route with destination internet and target nat gateway

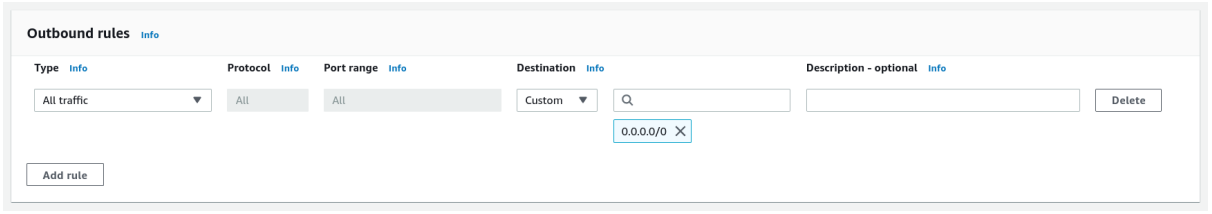


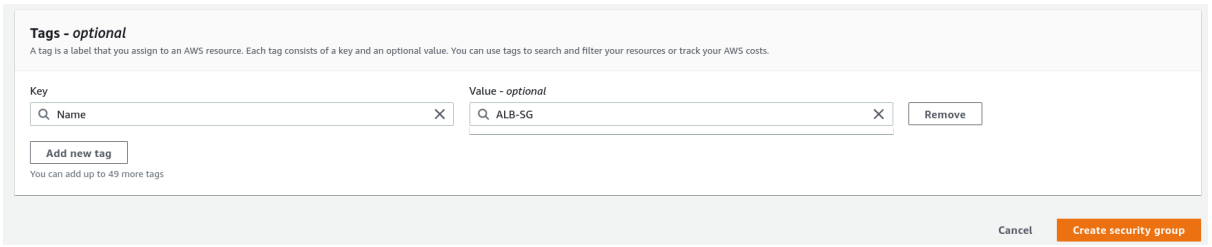
Step 7 : Create Security Groups

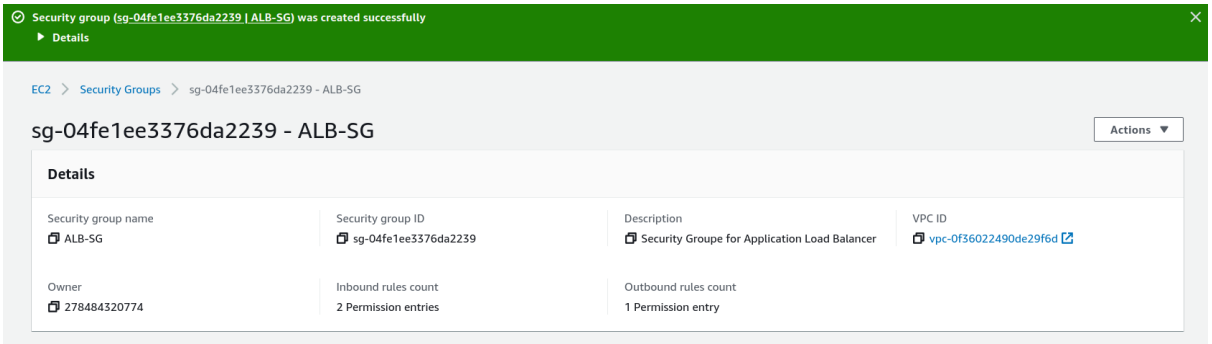
The security group will be attached to the Application load balancer. The security group needs to allow inbound HTTP and HTTPS traffic from 0.0.0.0/0 the public internet .



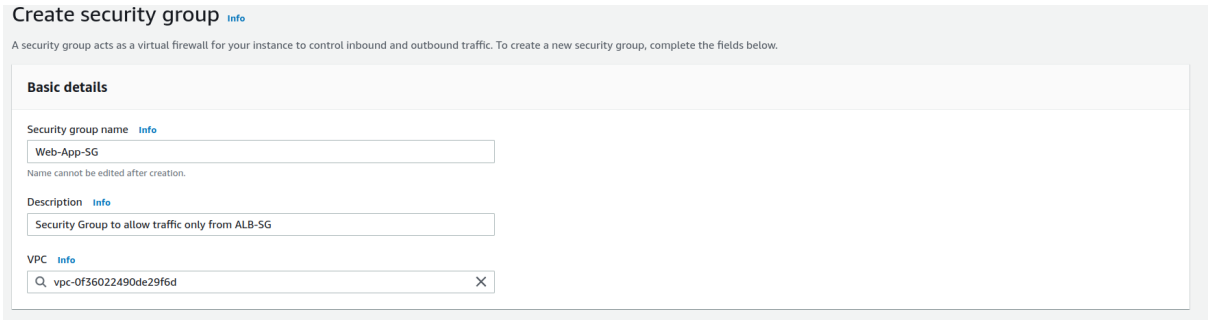


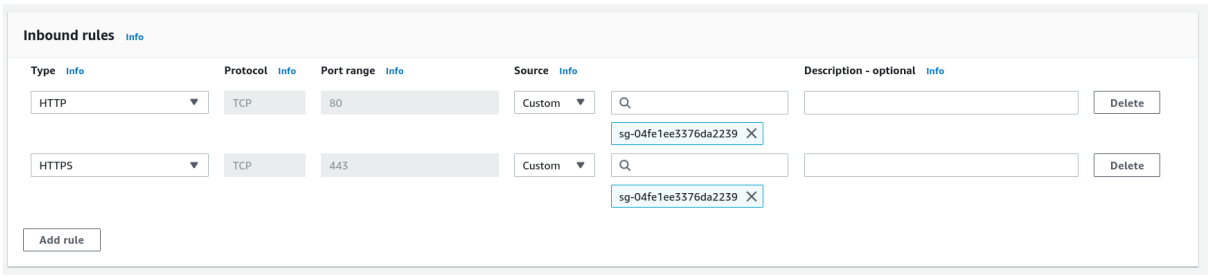


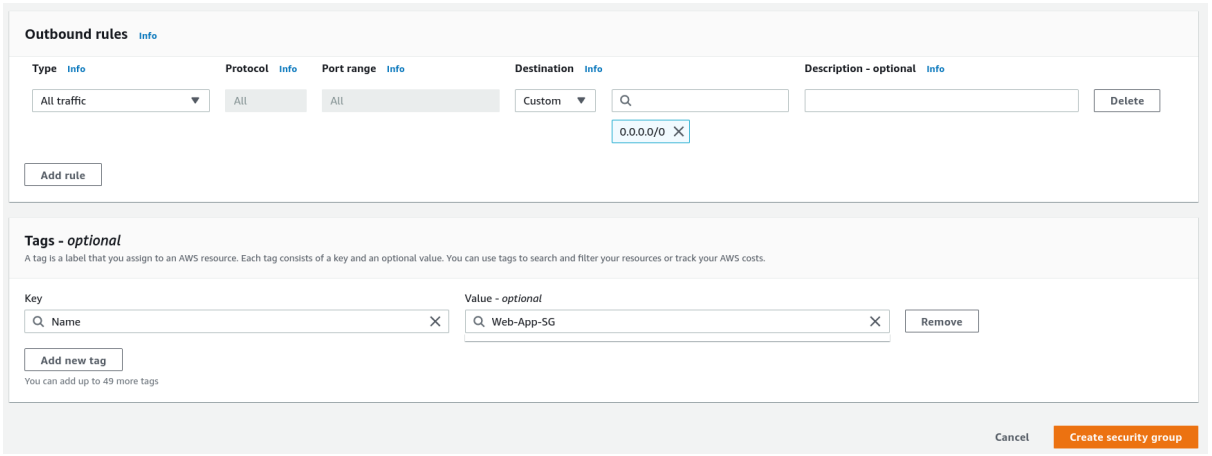


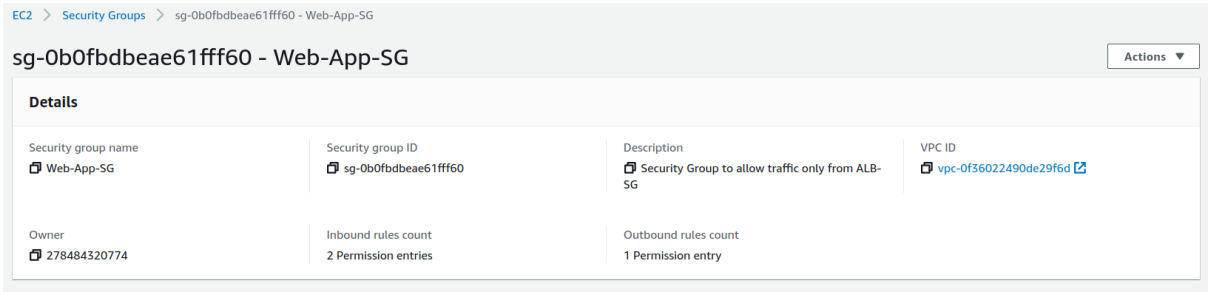


We need also to create a security group for instances that have our web app running on it .

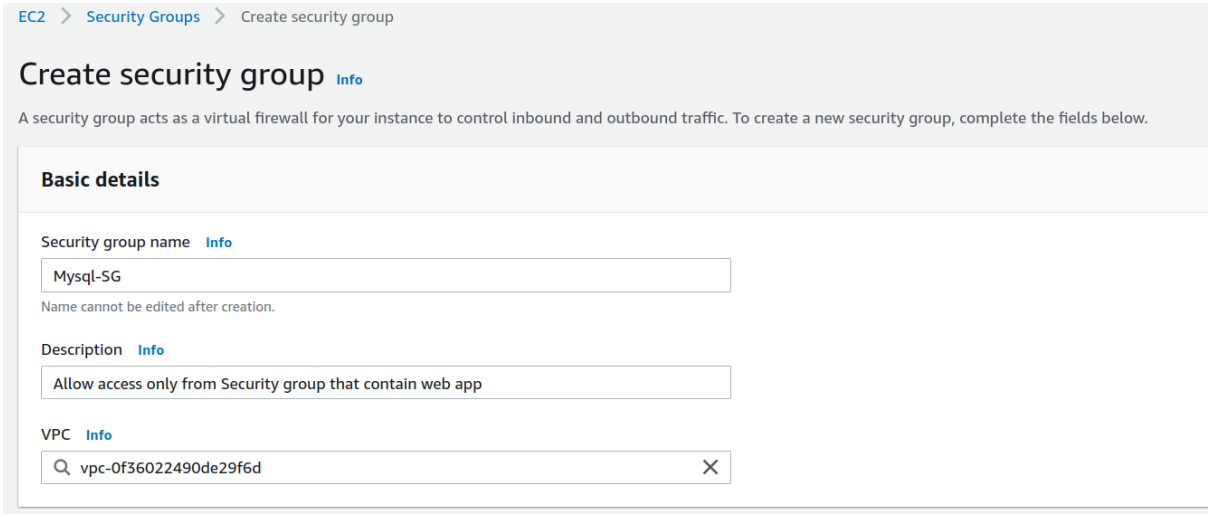


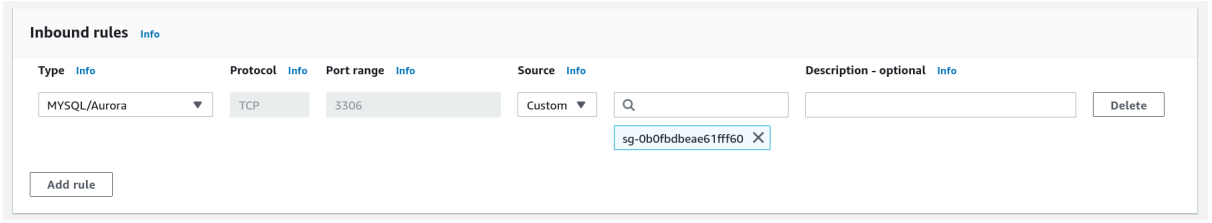


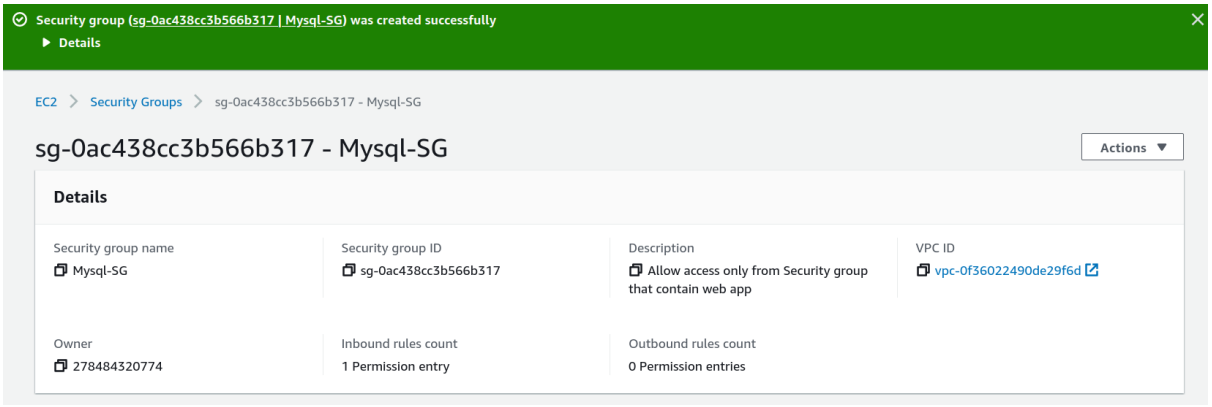




Also we need to create a security for db instance to allow only the traffic coming from Web-App-SG the security group of our web app .



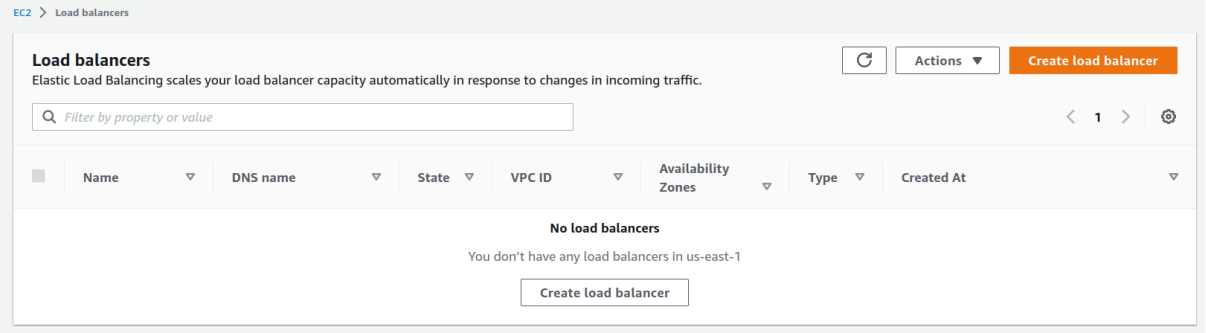




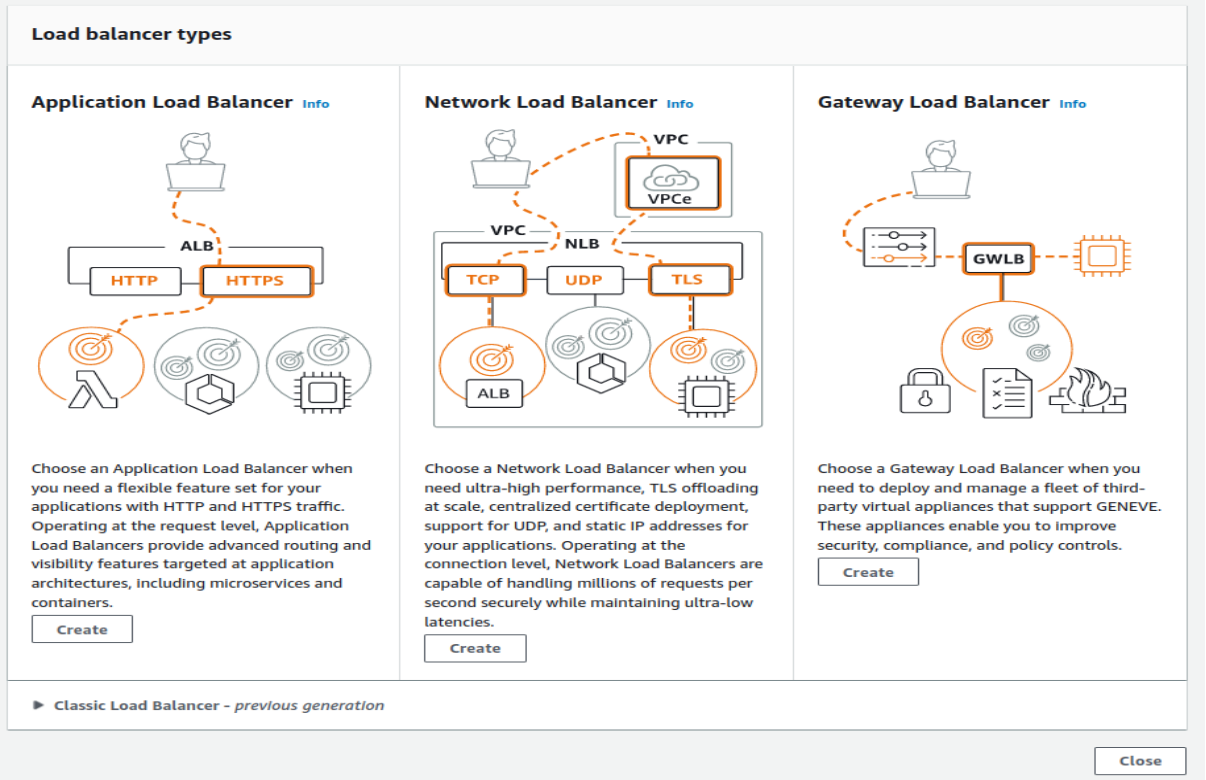
Step 8 : Create a LoadBalancer

Application Load Balancers distributes incoming application traffic across multiple targets, such as EC2 instances, in multiple Availability Zones. This increases the availability of your application.

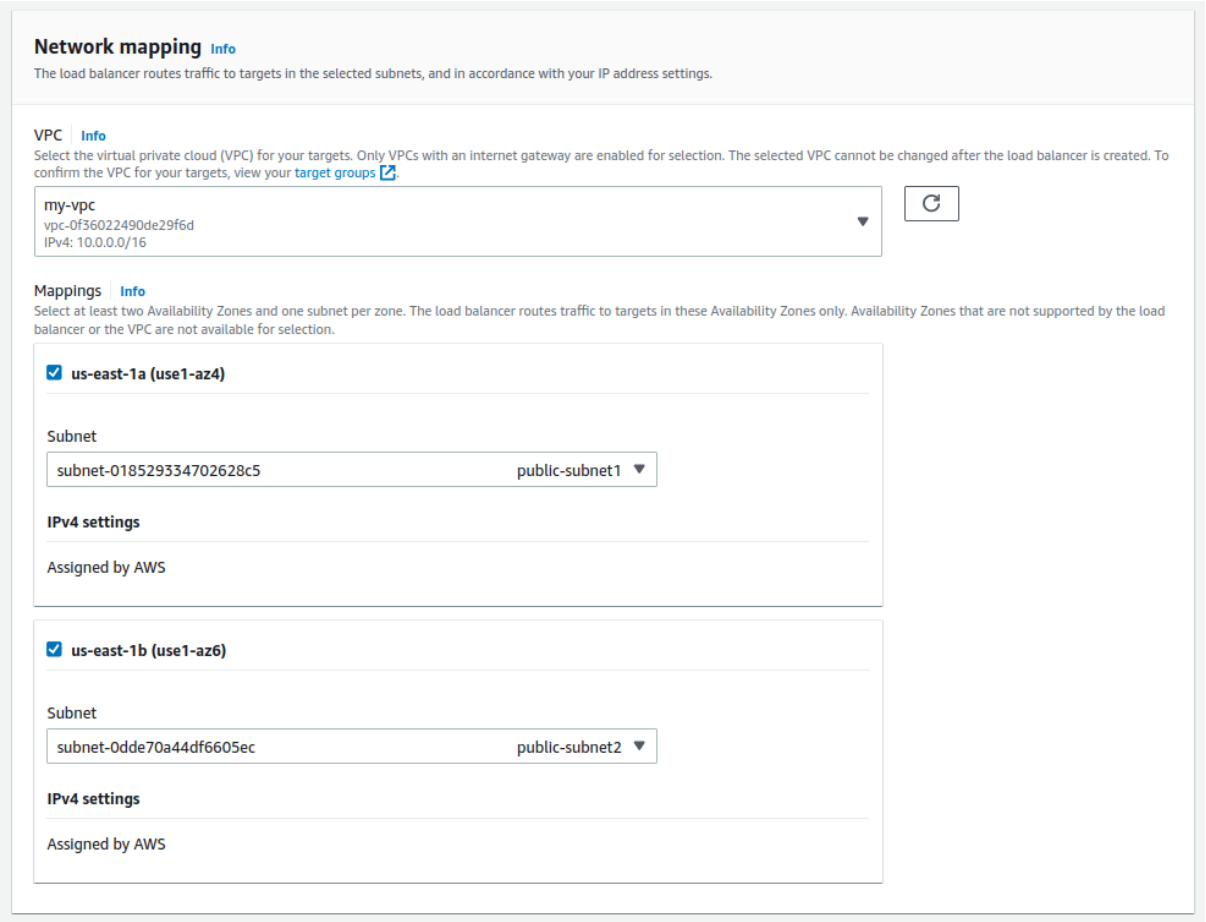
From the EC2 Dashboard, left side menu, scroll down and click *Load Balancers*, *Create Load Balancer*.



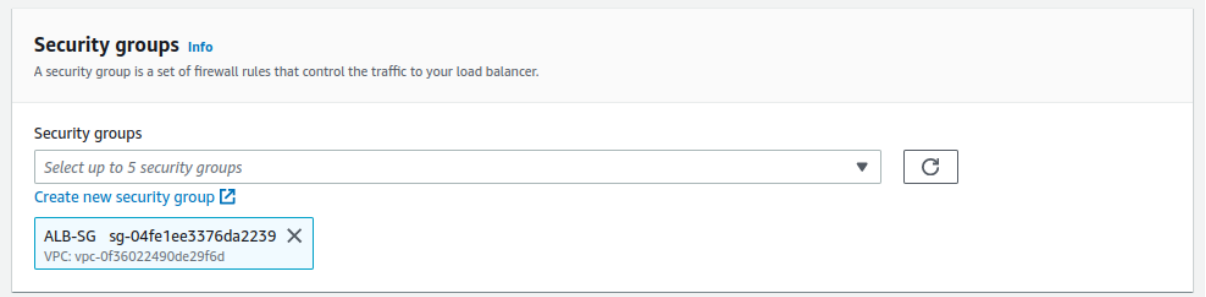
Name our ALB. Select *Internet facing*. Select *IPv4*. Select your VPC. Then We select each Subnet that We want the ALB to distribute the load between.



Choose the type Application load balancer and we make the Load Balancer span in two subnets



Select the security group for our load balancer to allow inbound traffic coming from the internet because our LB is internet facing not internal LB .

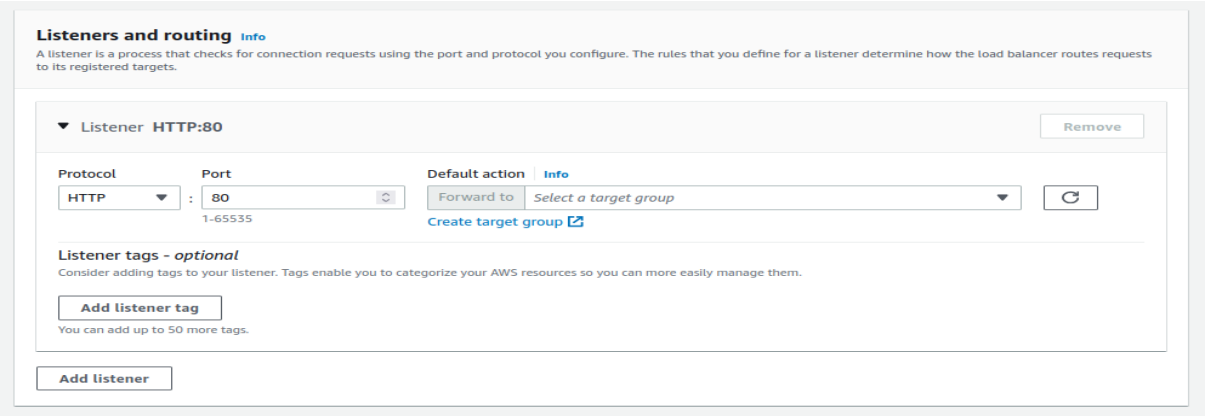


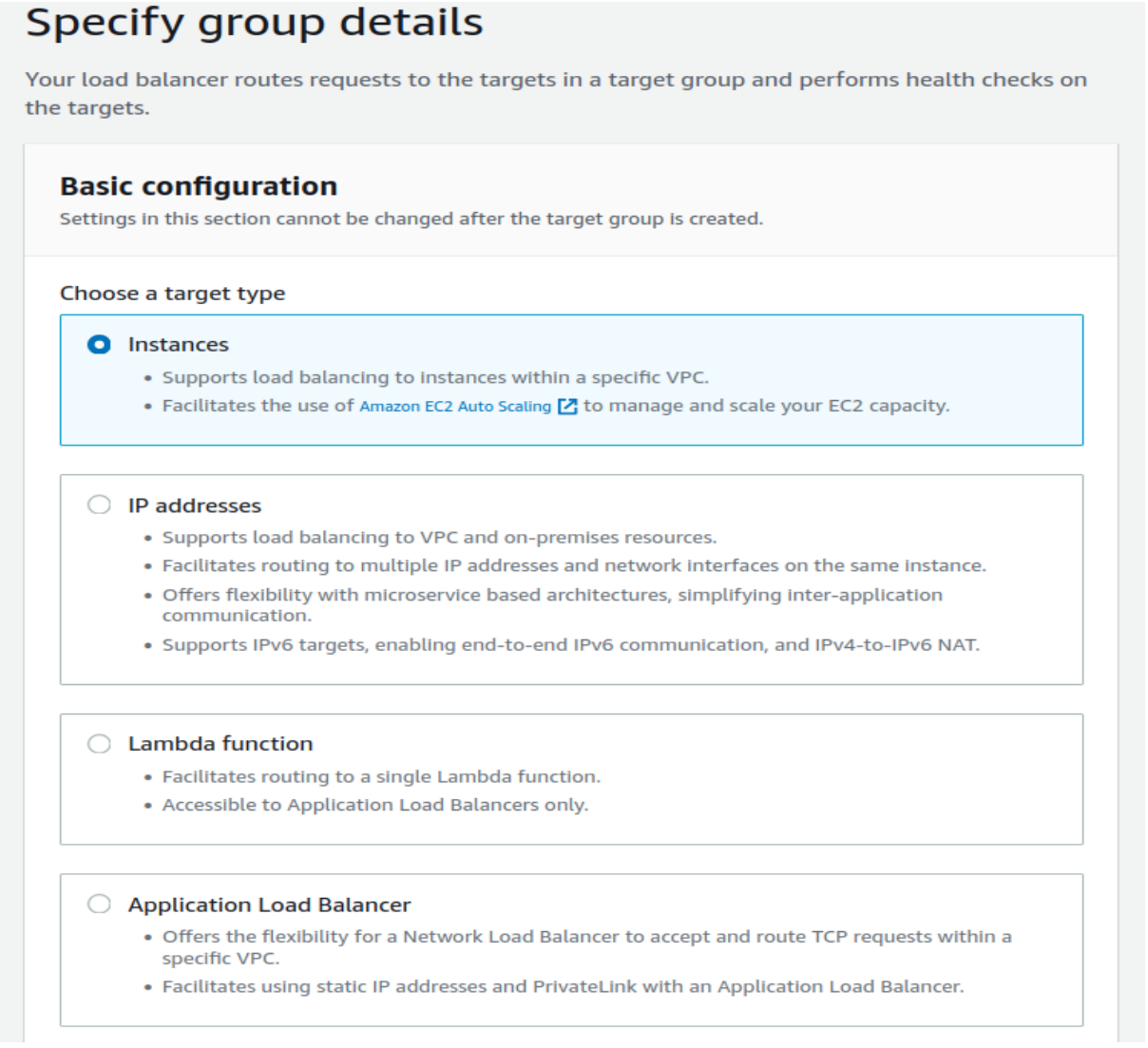
Scroll down and under *Listeners and routers* select *HTTP*, *port 80*, we do not not yet have a Target Group, so we will create one form here.

Each target group is used to route requests to one or more registered targets. When we create each listener rule, we specify a target group and conditions

Click *create target group*.

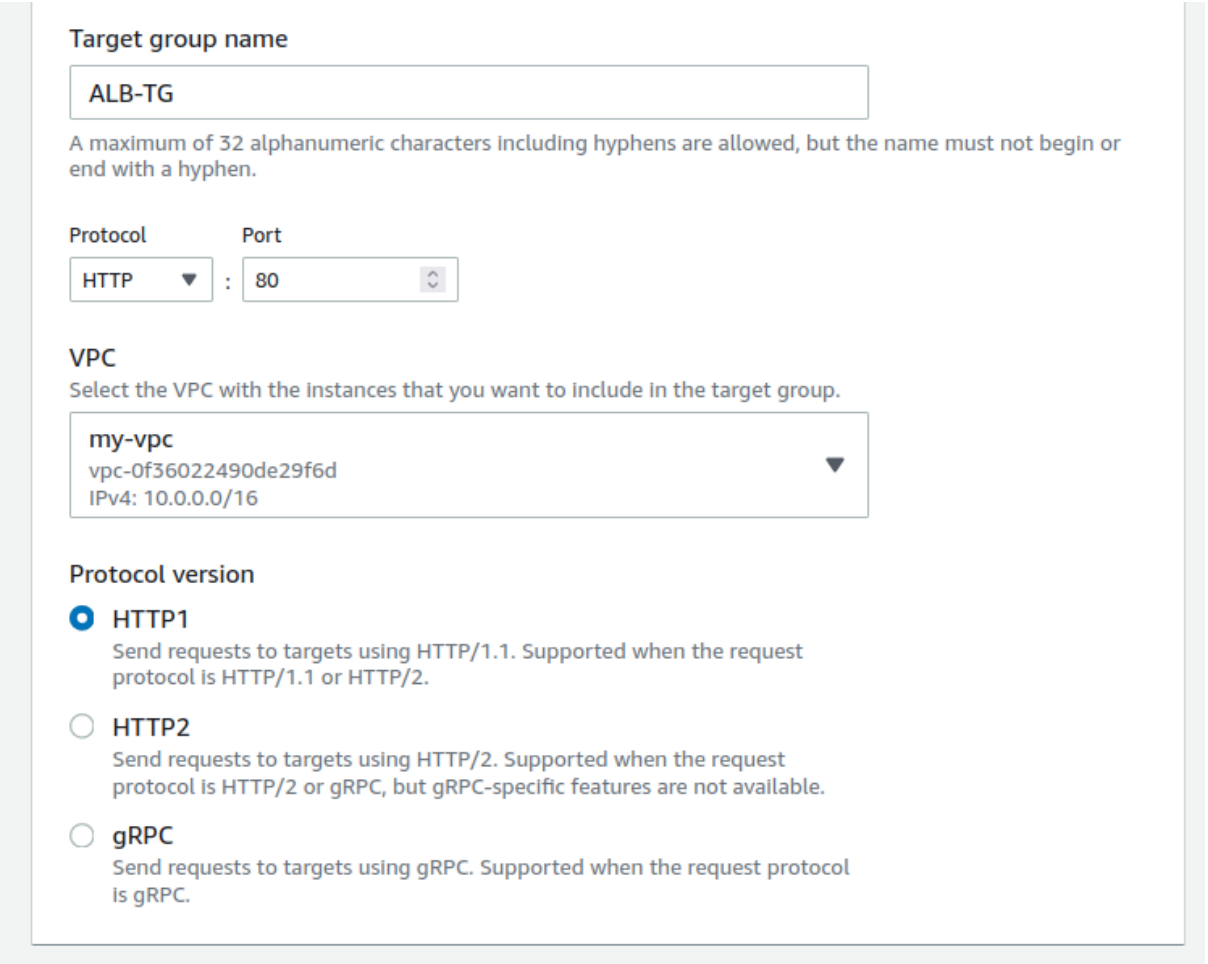
Choose your target type as *instances* and name your target group.

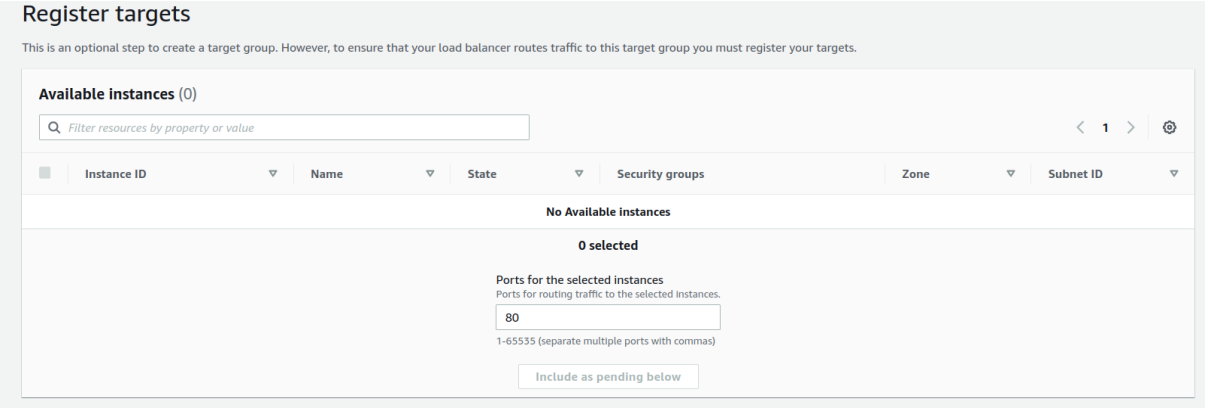


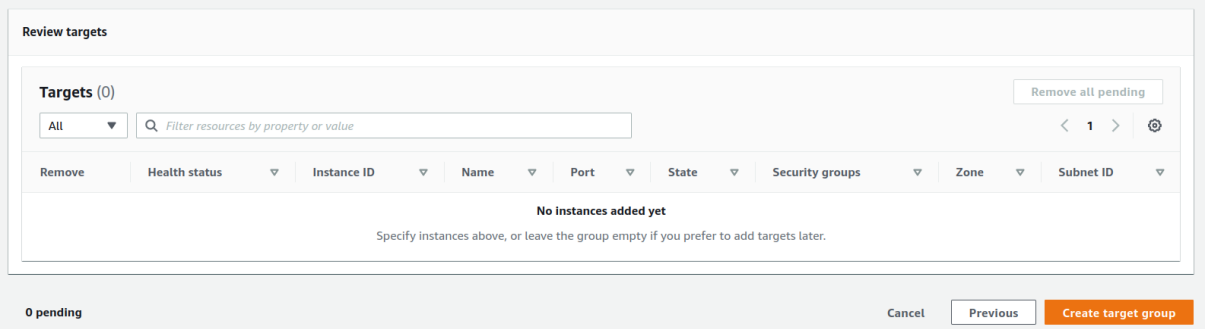


Give our target group a name and set the protocol to HTTP port 80.

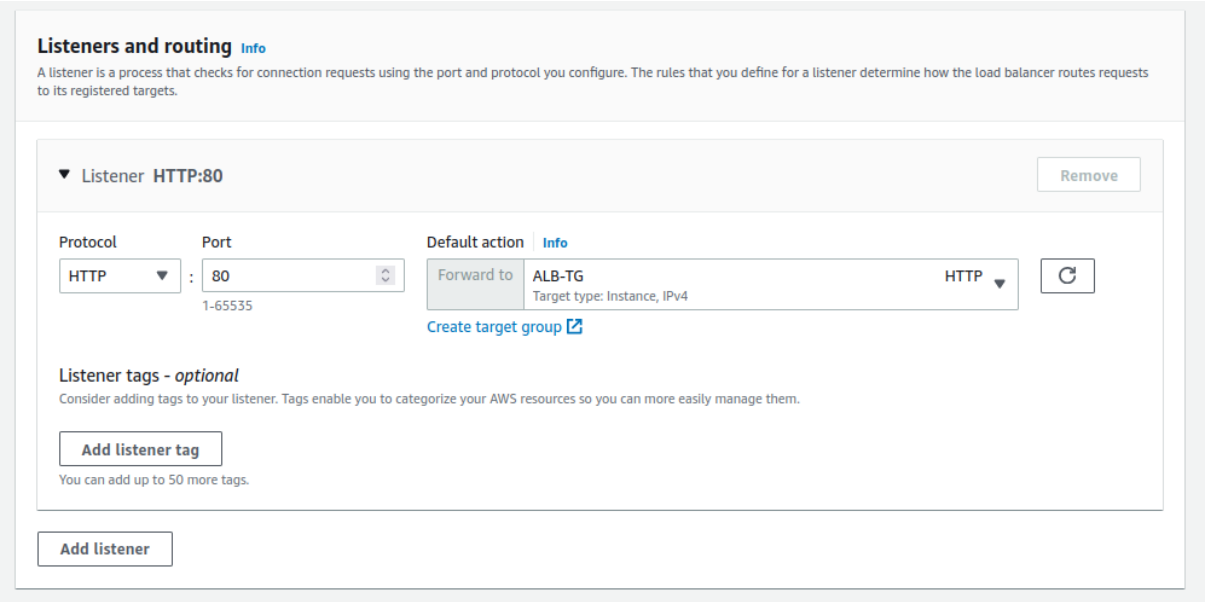
Select our VPC, select protocol version HTTP1 and click next.



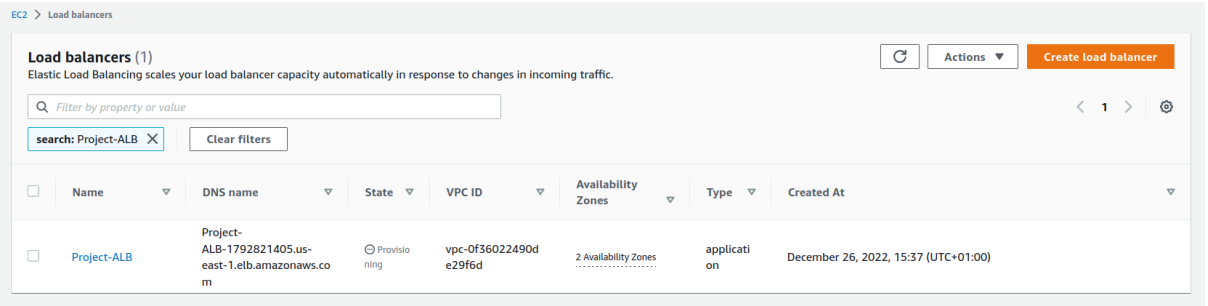




then click the button Create target group .
Now turn back to the screen to continue configuring our load balancer .



as we see our load balancer is created successfully as shown below

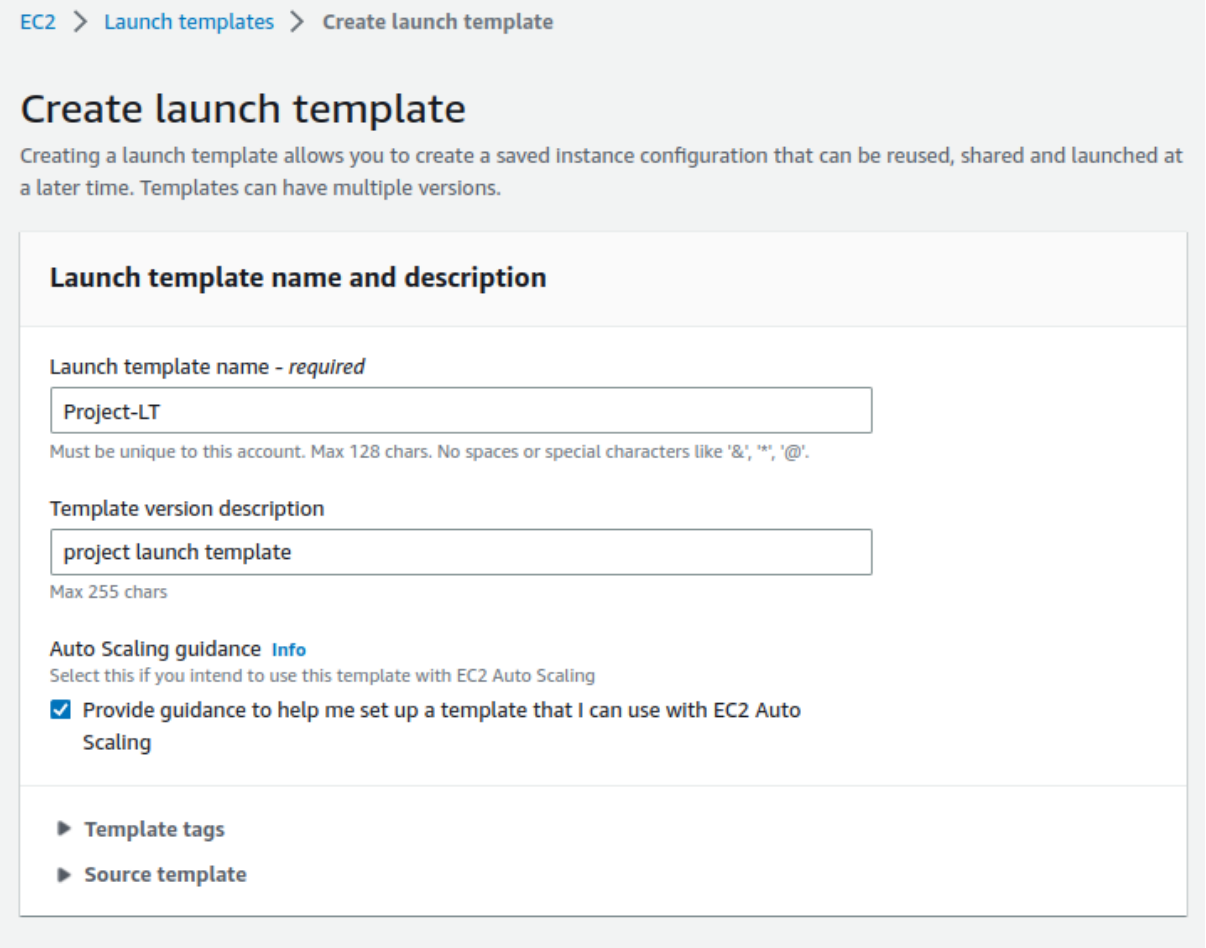


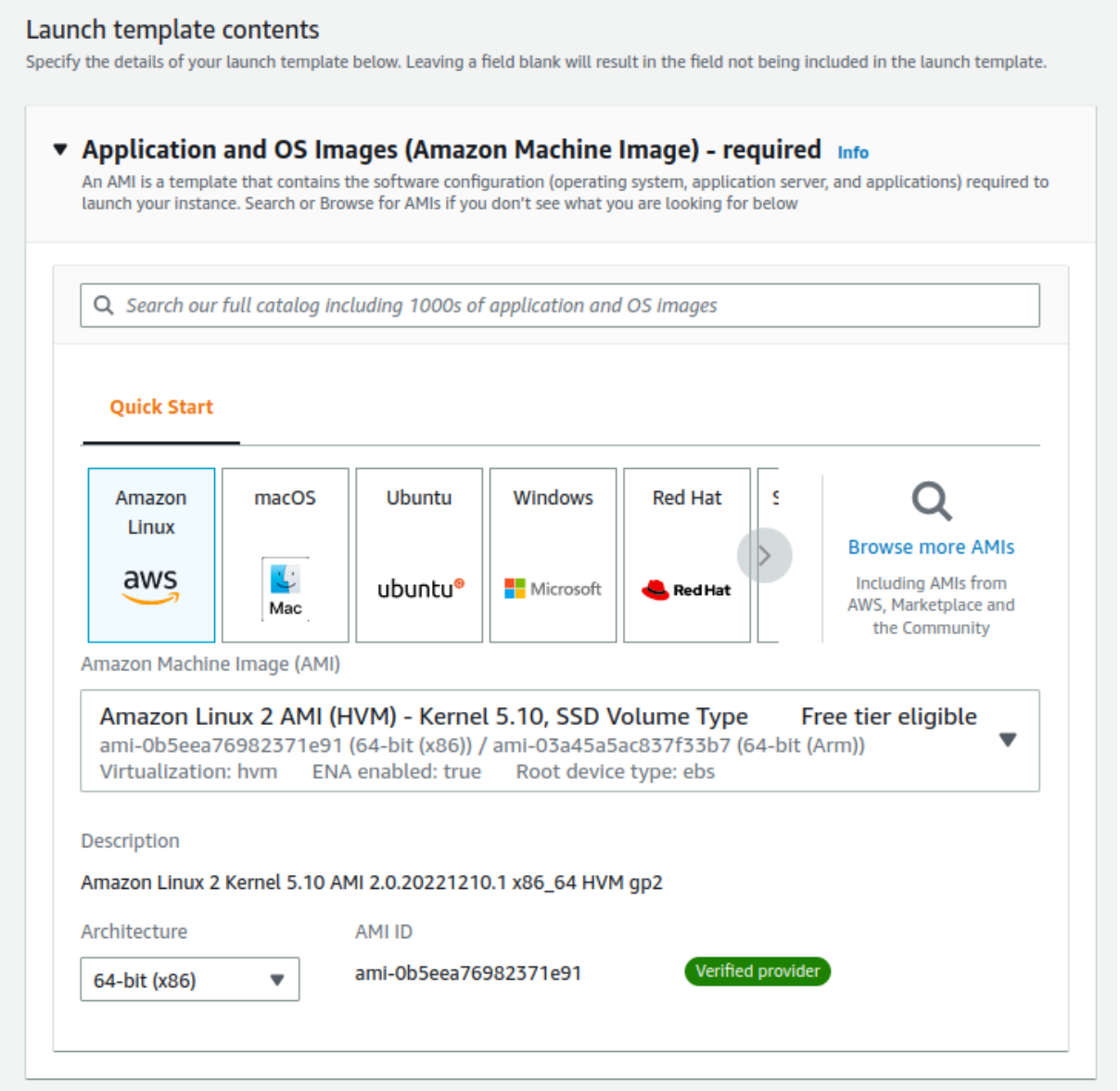
Step 9 : Create an autoscaling group

To help your instances adjust in response to demand, you have the option to create an auto scaling group. This can be done in the EC2 console via “Auto scaling groups” on the left side menu. When creating the auto scaling group, you must first choose a launch template, which is a saved instance configuration that includes an AMI, instance type, key pair, security group, and any user data that you want to be executed upon launch of the instance. If you don’t already have a launch template, you should create a new one prior to creating the auto scaling group. The launch template creation is a bit lengthy, but starts with selecting “Launch Templates” on the left side menu in the EC2 console.

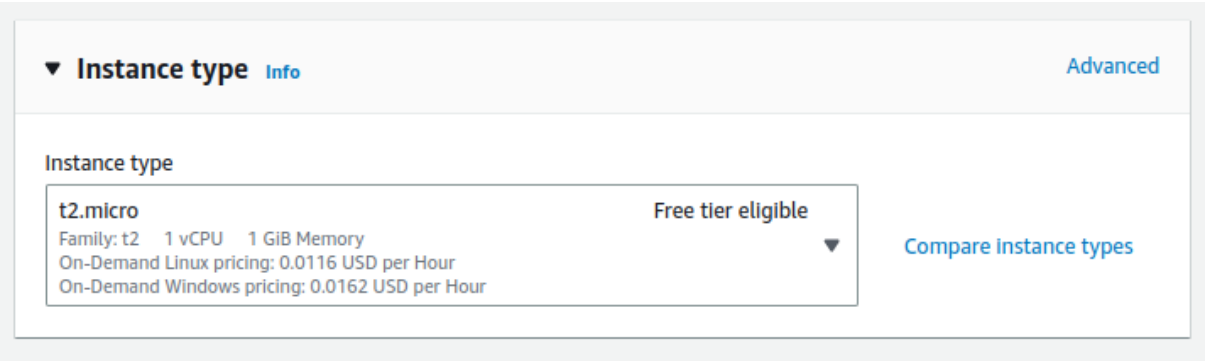
First let’s create a launch template :

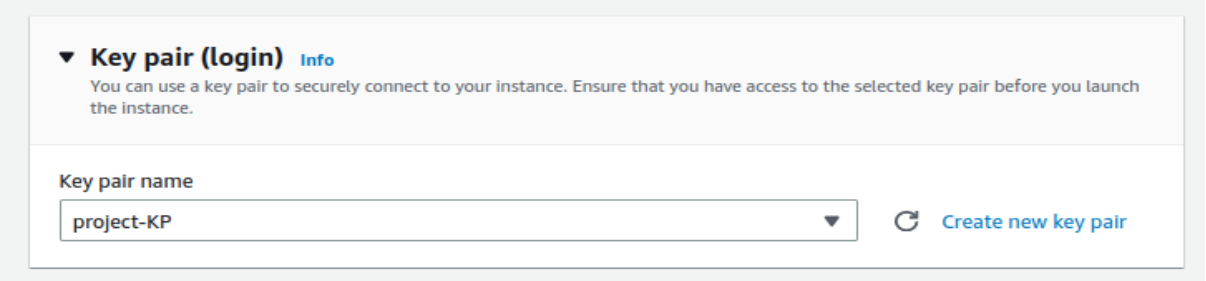
Click the orange “**Create launch template**” button and get started by choosing a launch template name and selecting an AMI. Below I have selected an Amazon Linux 2 AMI with 64 bit architecture.

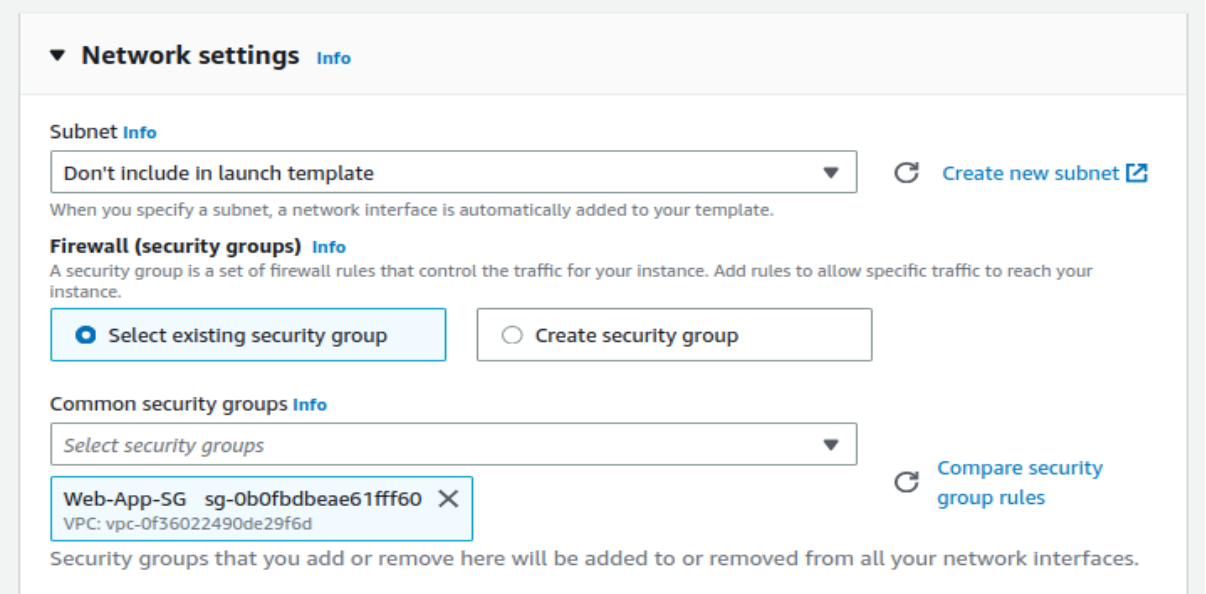




Next, I've selected a t2.micro instance type, a key pair (you can create a new key pair if you don't already have one), and I've selected the security group I created earlier. Because this launch template will be used for an auto scaling group, we do not need to include a subnet here.







Select “Enable” under “Auto-assign public IP.” The next sections are “Storage (volumes)” and “Resource tags” and we have chosen not to change any of those default settings.

Storage (volumes) Info

EBS Volumes Hide details

Volume 1 (AMI Root) (8 GiB, EBS, General purpose SSD (gp2))
AMI Volumes are not included in the template unless modified

Add new volume

Resource tags Info

No resource tags are currently included in this template. Add a resource tag to include it in the launch template.

Add tag

50 remaining (Up to 50 tags maximum)

Advanced details Info

Advanced network configuration

Network interface 1 Remove

Device index Info: 0

Network interface Info: New interface

Description Info:

Subnet Info: Don't include in launch template
Not applicable for EC2 Auto Scaling

Security groups Info: Select security groups
Show all selected (1)

Auto-assign public IP Info: Enable

Primary IP Info:
Not applicable for EC2 Auto Scaling

Secondary IP Info: Don't include in launch tem...
Not applicable for EC2 Auto Scaling

IPv6 IPs Info: Don't include in launch tem...
Not applicable for EC2 Auto Scaling

IPv4 Prefixes Info: Don't include in launch tem...
The selected instance type does not support IPv4 prefixes.

IPv6 Prefixes Info: Don't include in launch tem...
The selected instance type does not support IPv6 prefixes.

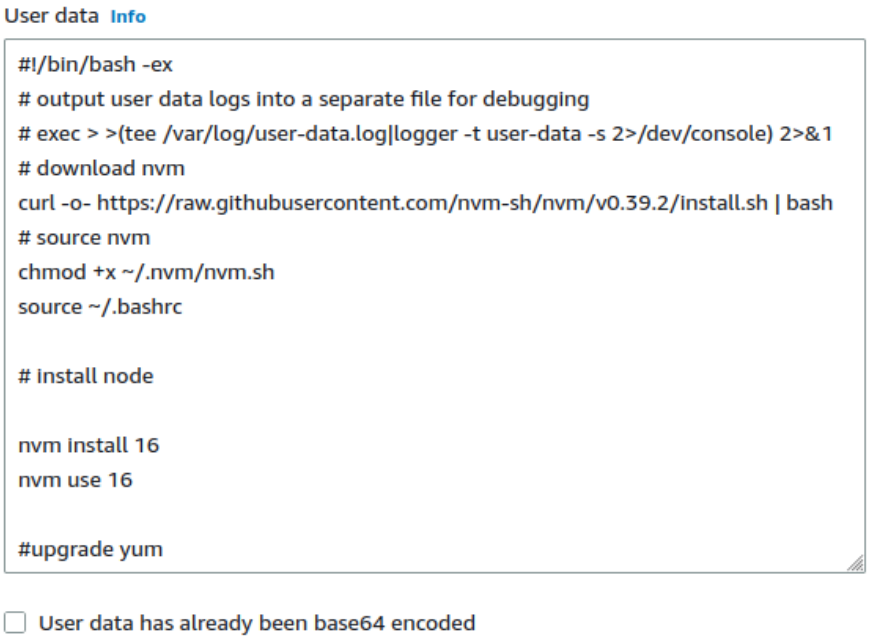
Delete on termination Info: Don't include in launch tem...

Elastic Fabric Adapter Info
Enable
EFA is only compatible with certain instance types.

Network card index Info: Don't include in launch tem...
The selected instance type does not support multiple network cards.

Add network interface

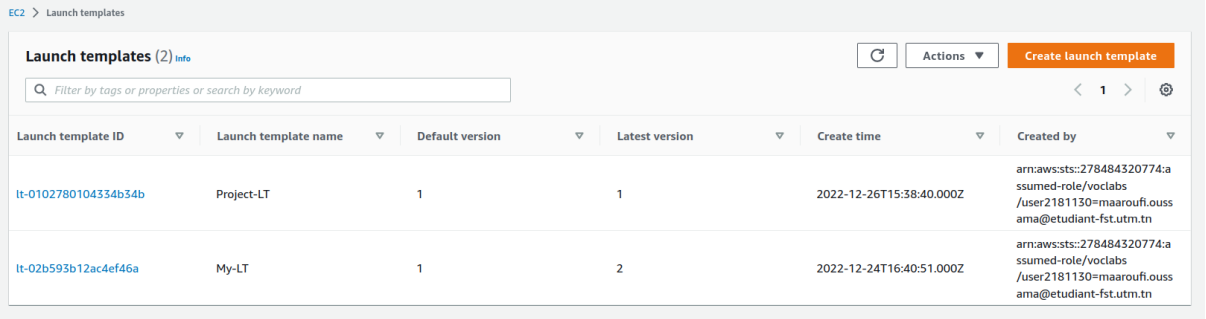
In the advanced section we add a user_data script that will allow us to deploy our express backend after installing all the necessary dependencies .



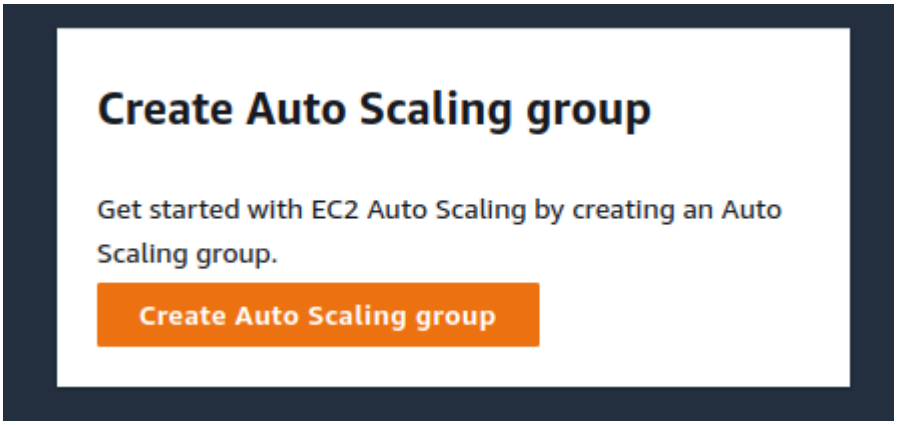
This is the complete script

```
#!/bin/bash
# download nvm
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.2/install.sh | bash
# source nvm
chmod +x ~/.nvm/nvm.sh
source ~/.bashrc

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"
# install node
nvm install 16
nvm use 16
#upgrade yum
sudo yum upgrade
#install git
sudo yum install git -y
cd /home/ec2-user
# get source code from githubt
git clone https://github.com/OussamaMaroufi/Express-Backend.git
#get in server dir
cd Express-Backend
#give permission
sudo chmod -R 777 .
#install node module
npm install
# start the app
npm run dev > app1.out.log 2> app1.err.log < /dev/null &
# to redirect traffic comes to port 80 into port 3001
sudo iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 3001
```

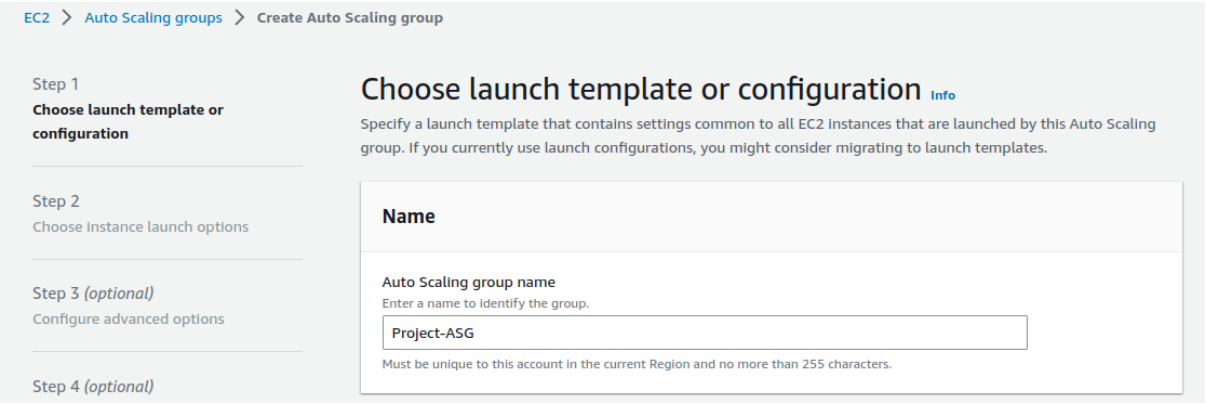
When we finish configuring the launch template, click “Create launch template.”

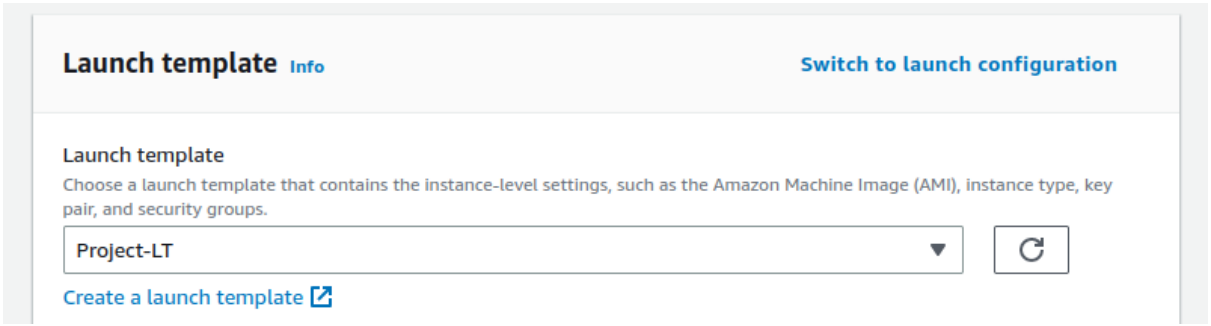


Now we are ready to create the Auto Scaling Group.On the EC2 dashboard we scroll down and select *auto scaling groups*, *create auto scaling groups*.

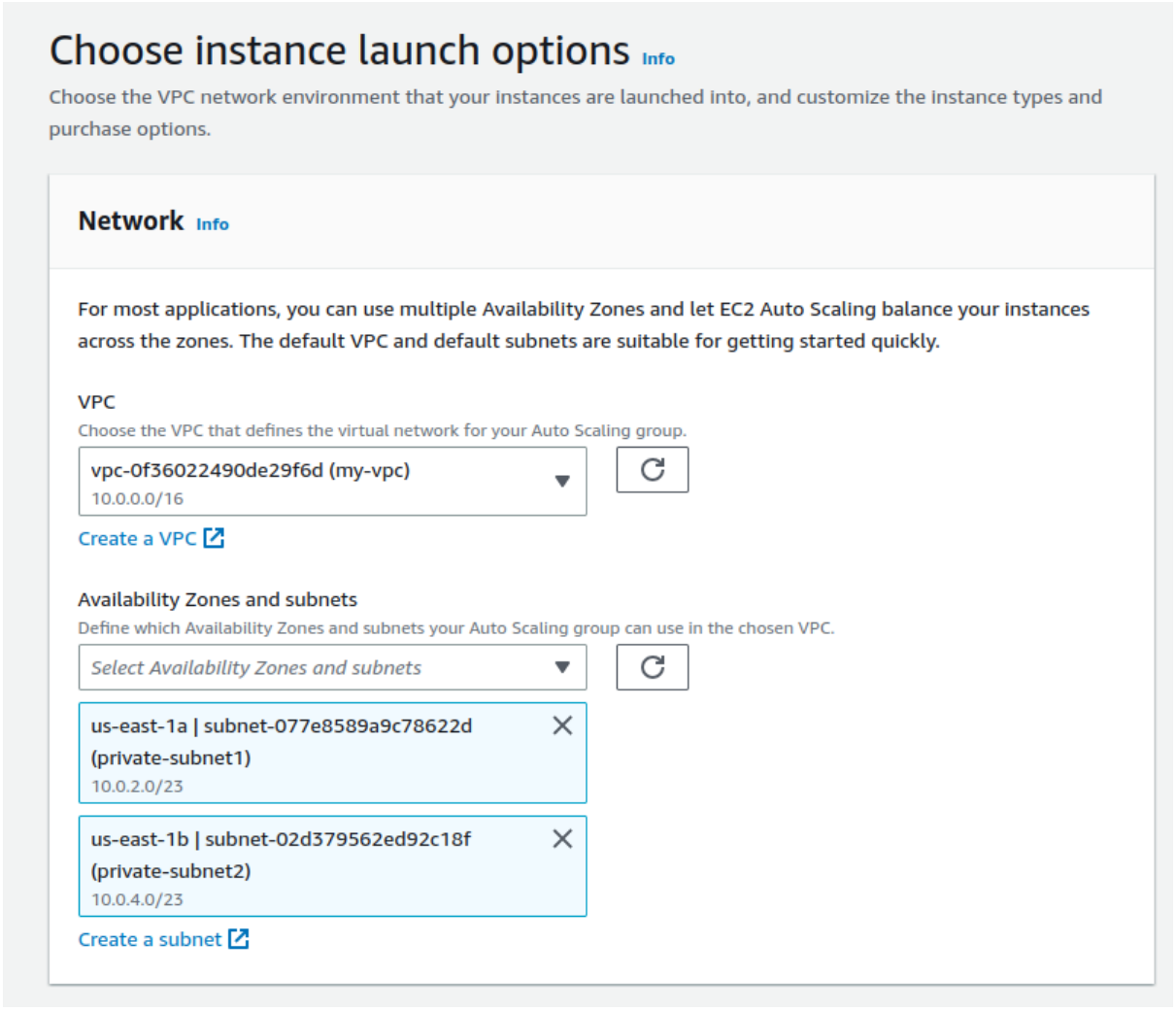


Give our auto scaling group a name and select the launch template we just created. Scroll down and *click next*.



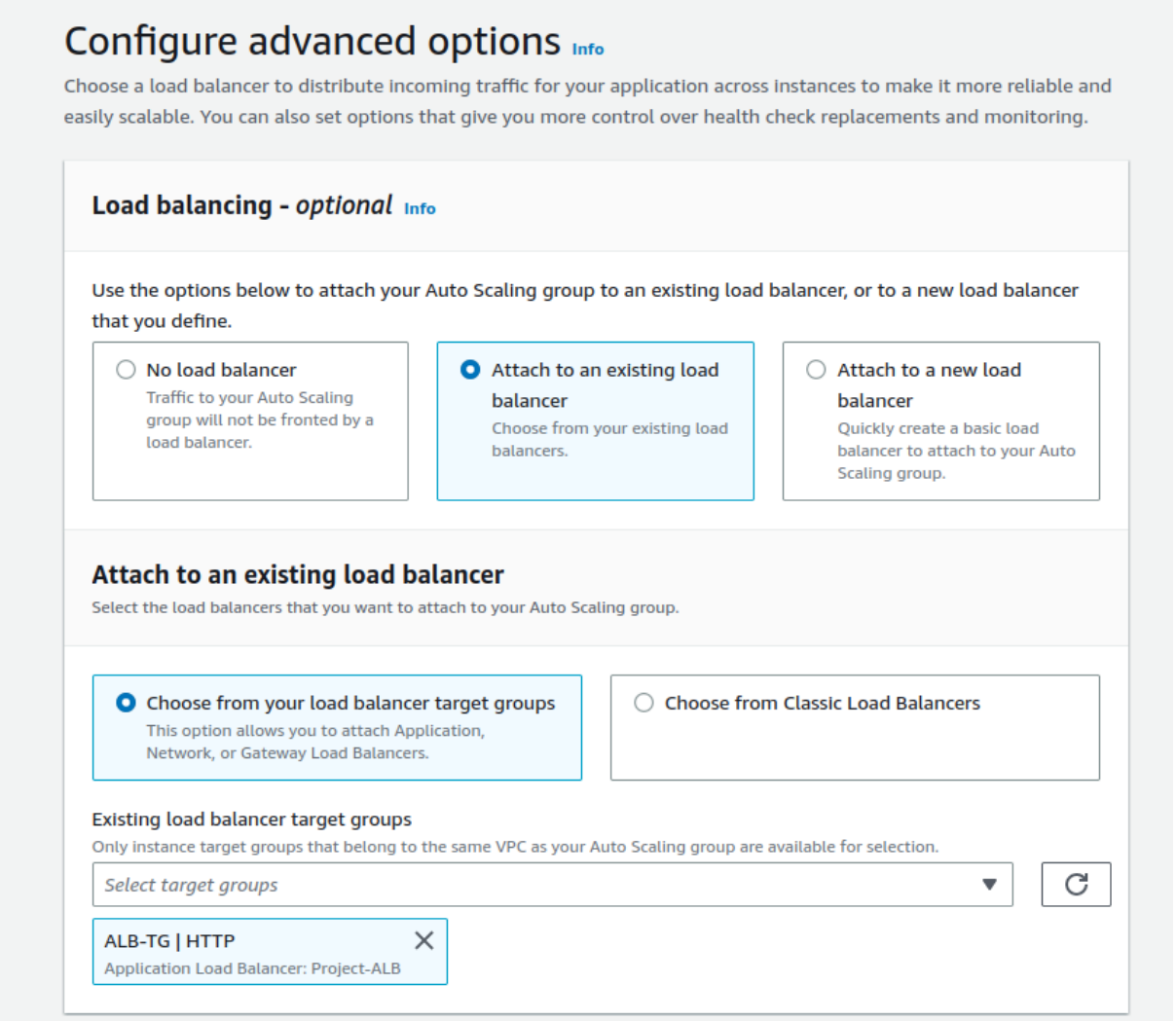


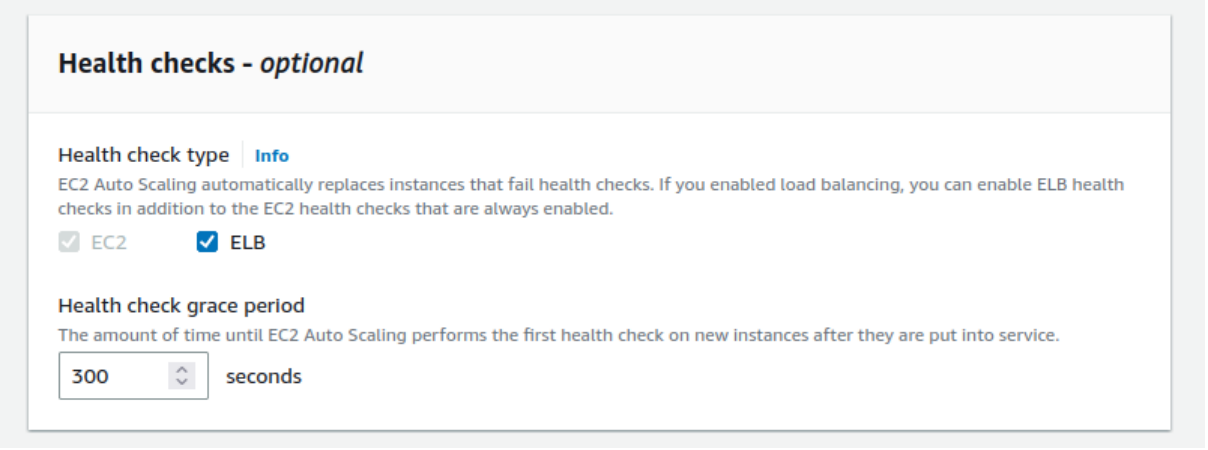
Select our custom VPC and subnets. *Click next*.



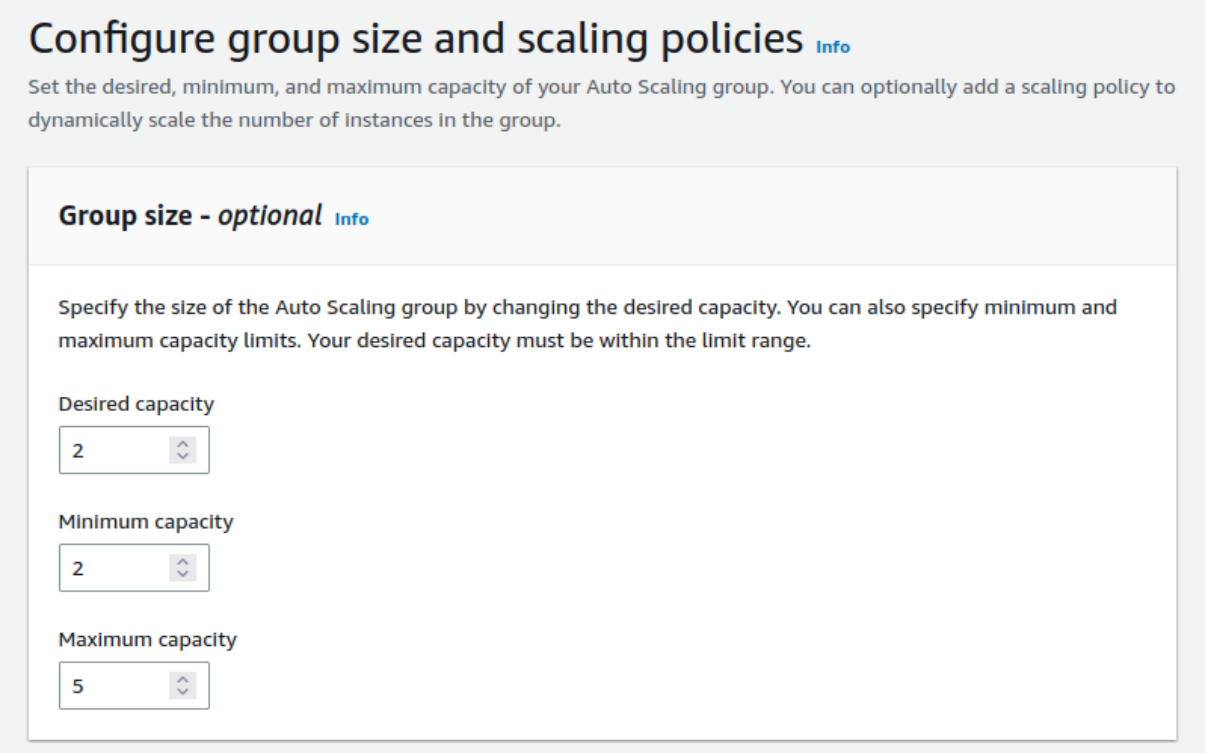
click next.

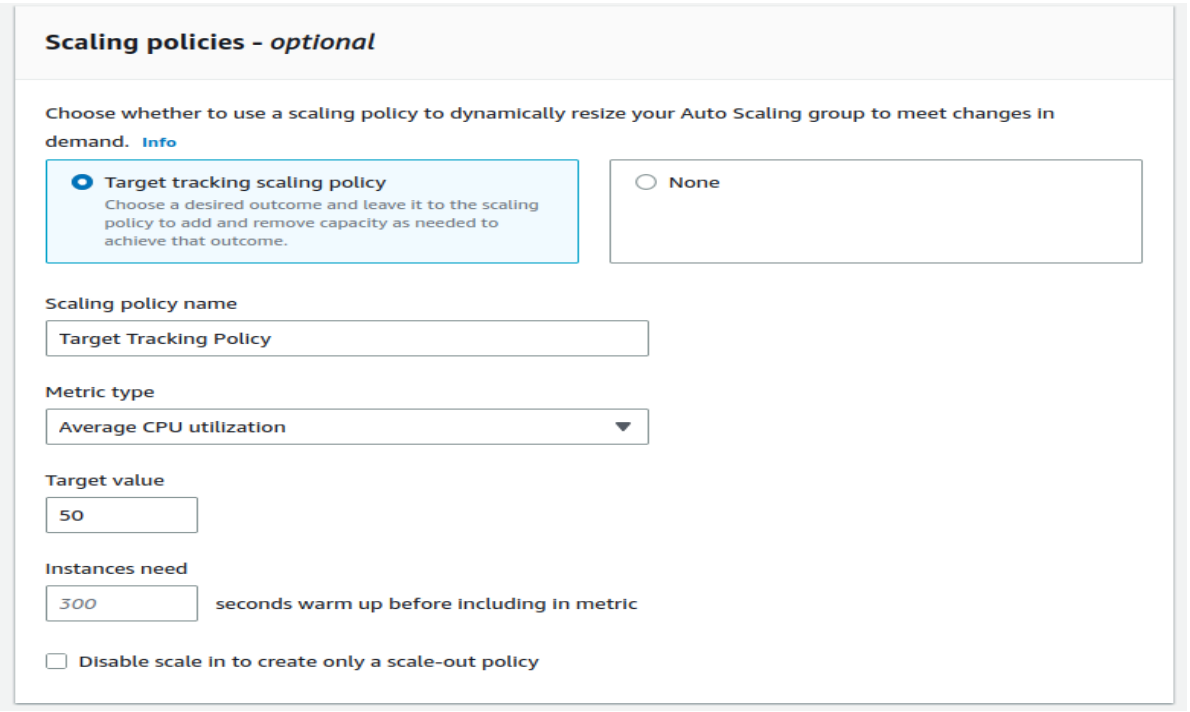
Select *Attach to existing load balancer* and the target group that we have already created, under health check type select *ELB,* select *next.*



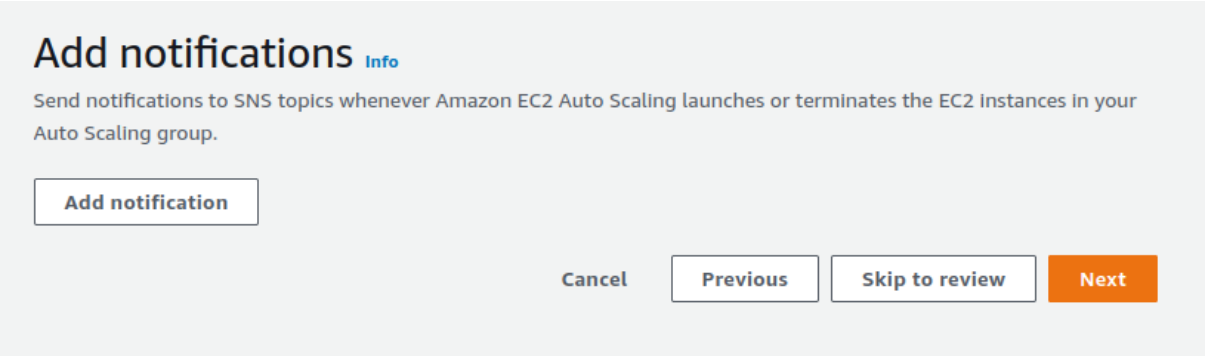


We will instruct our Auto Scaling Group to always ensure we have a minimum of two instances running at a time. Configure the following scaling policy.

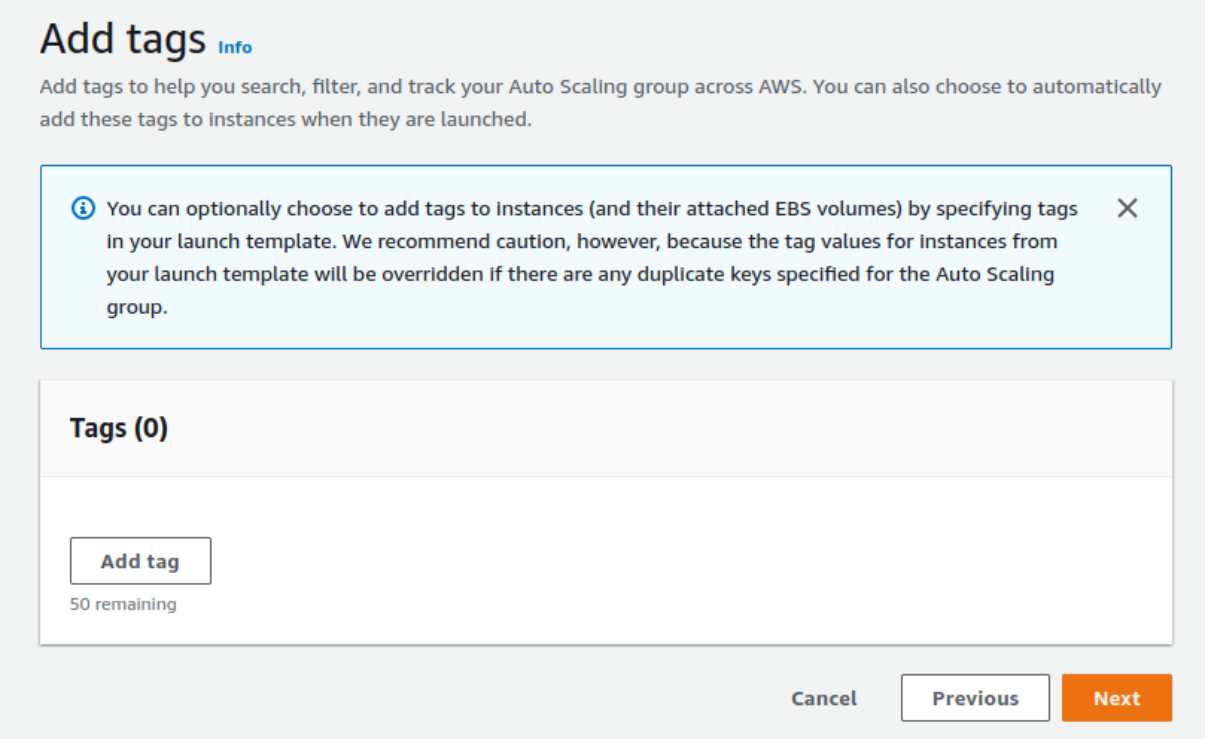




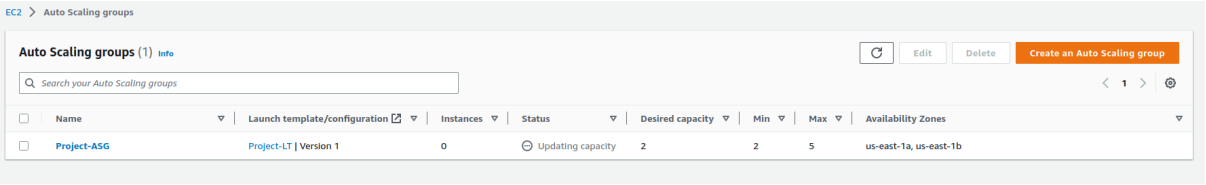
click Next we don’t need to add Notification to our autoscaling group for now .



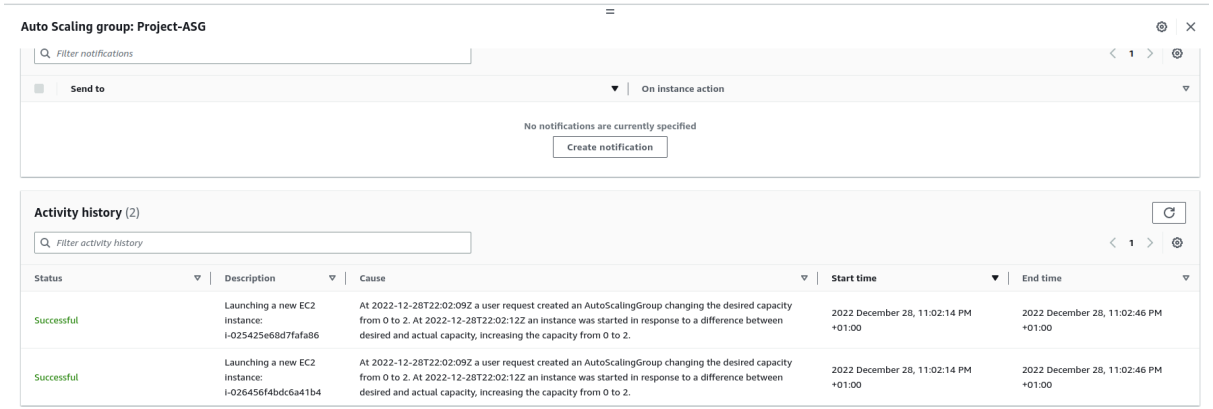
click Next



the click create auto scaling group

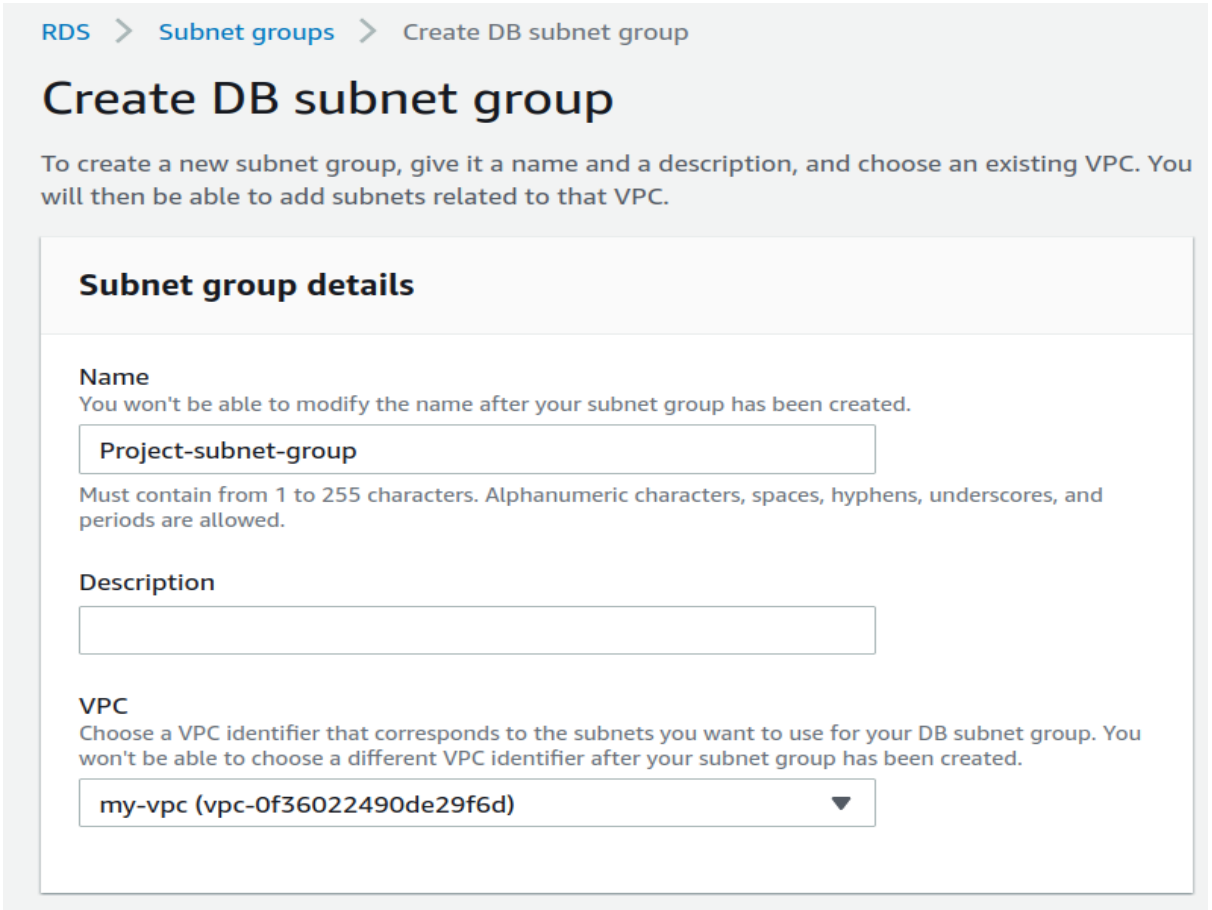


if we check the activity history we see that our instances are successfully launched



Step 10 : Create Database Instance RDS - MYSQL

First, set up the MySQL database by creating subnet groups for this database. Navigate to RDS > Subnet Groups > Create DB subnet group.
“A DB subnet group is a collection of subnets (typically private) that you create for a VPC and that you then designate for your DB instances.”



Add subnets

Availability Zones
Choose the Availability Zones that include the subnets you want to add.

Choose an availability zone

us-east-1a ✕ us-east-1b ✕

Subnets
Choose the subnets that you want to add. The list includes the subnets in the selected Availability Zones.

Select subnets

us-east-1a

- [ ] subnet-018529334702628c5 (10.0.0.0/24)
- [x] subnet-077e8589a9c78622d (10.0.2.0/23)

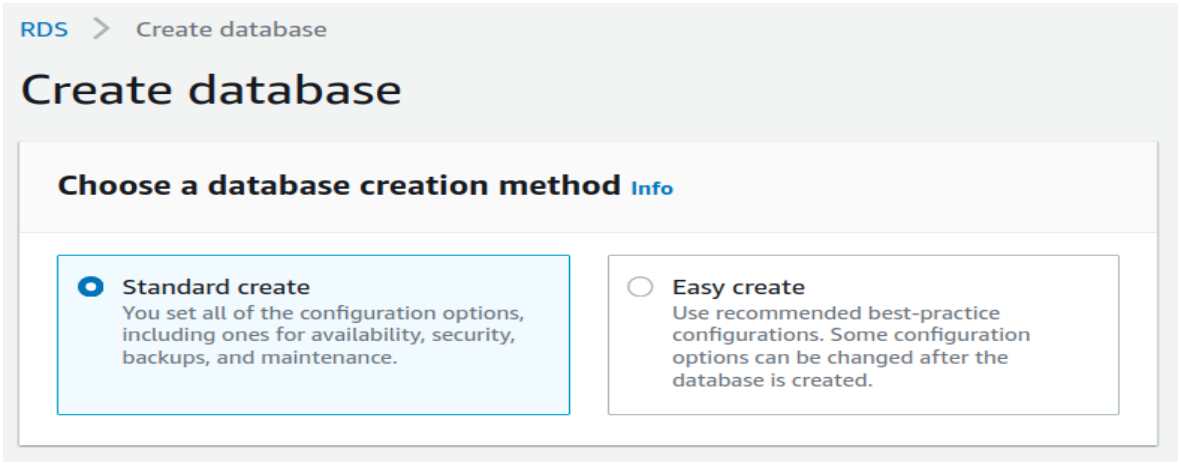
us-east-1b

- [ ] subnet-0dde70a44df6605ec (10.0.1.0/24)
- [x] subnet-02d379562ed92c18f (10.0.4.0/23)

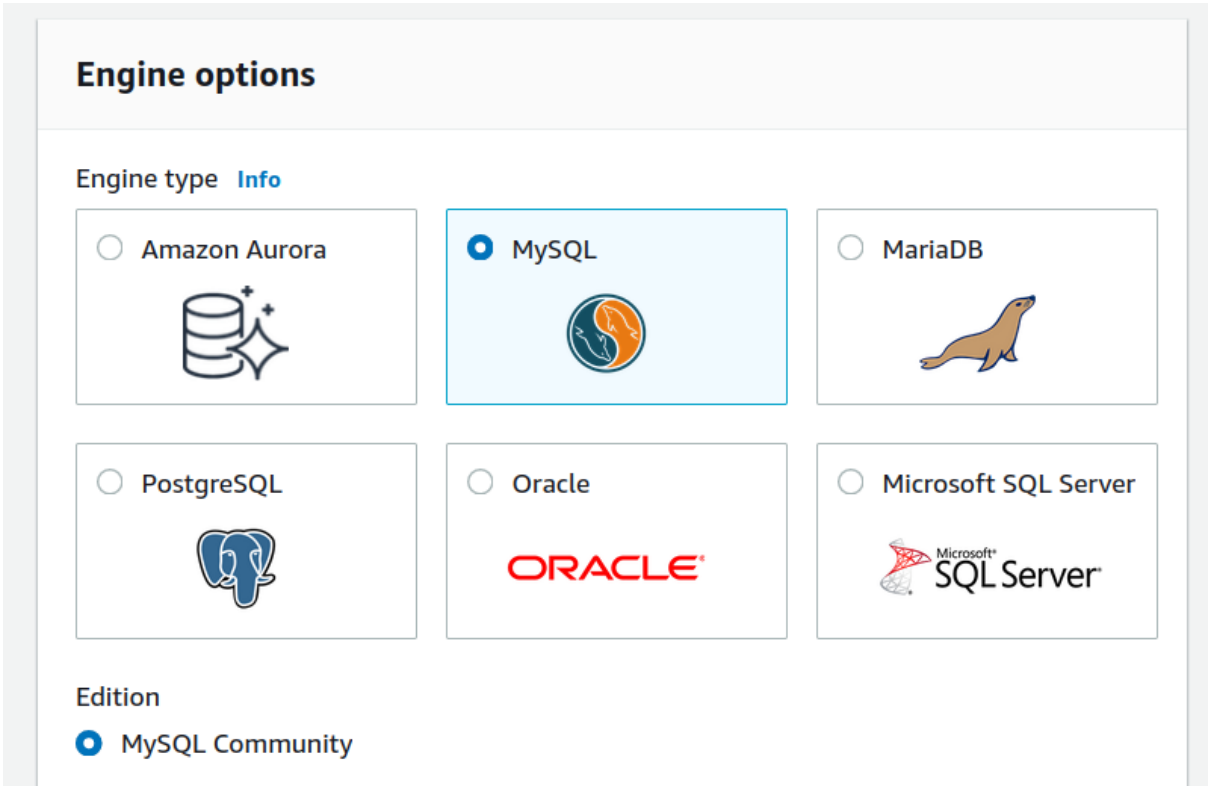
		CIDR block
us-east-1a	subnet-077e8589a9c78622d	10.0.2.0/23
us-east-1b	subnet-02d379562ed92c18f	10.0.4.0/23

Cancel Create

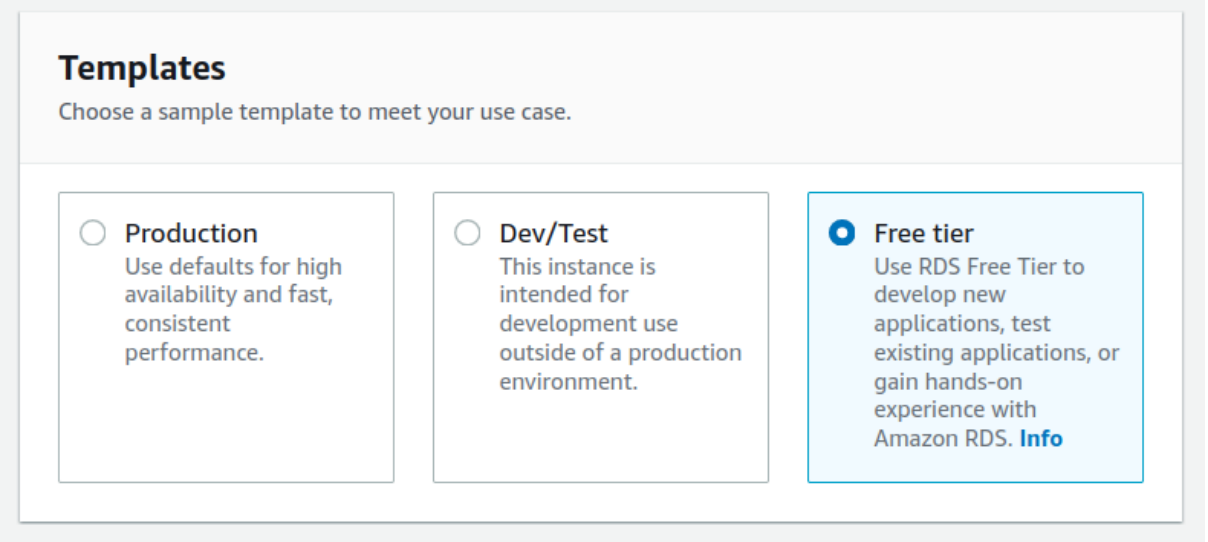
Next, we navigate back to the RDS dashboard to create the MySQL database.
We are using a Standard create with MySQL selected as the engine option.

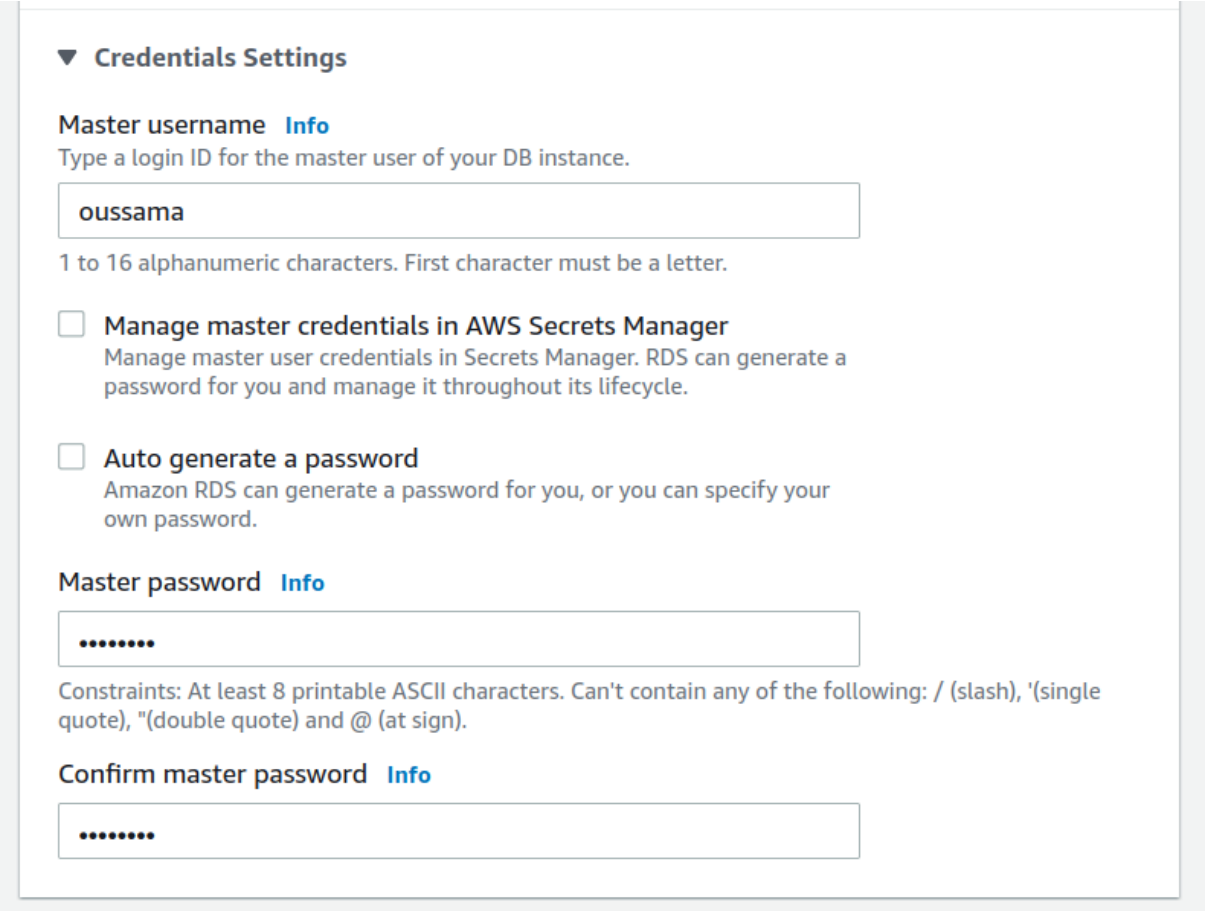


Select Mysql Engine

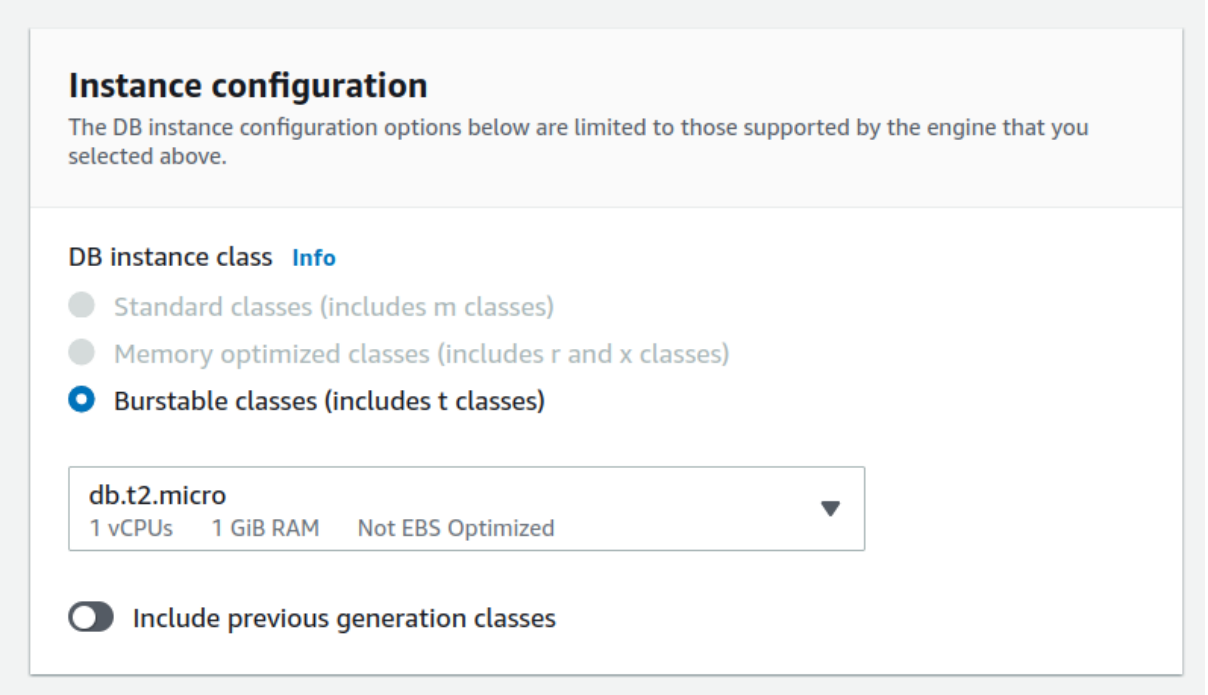


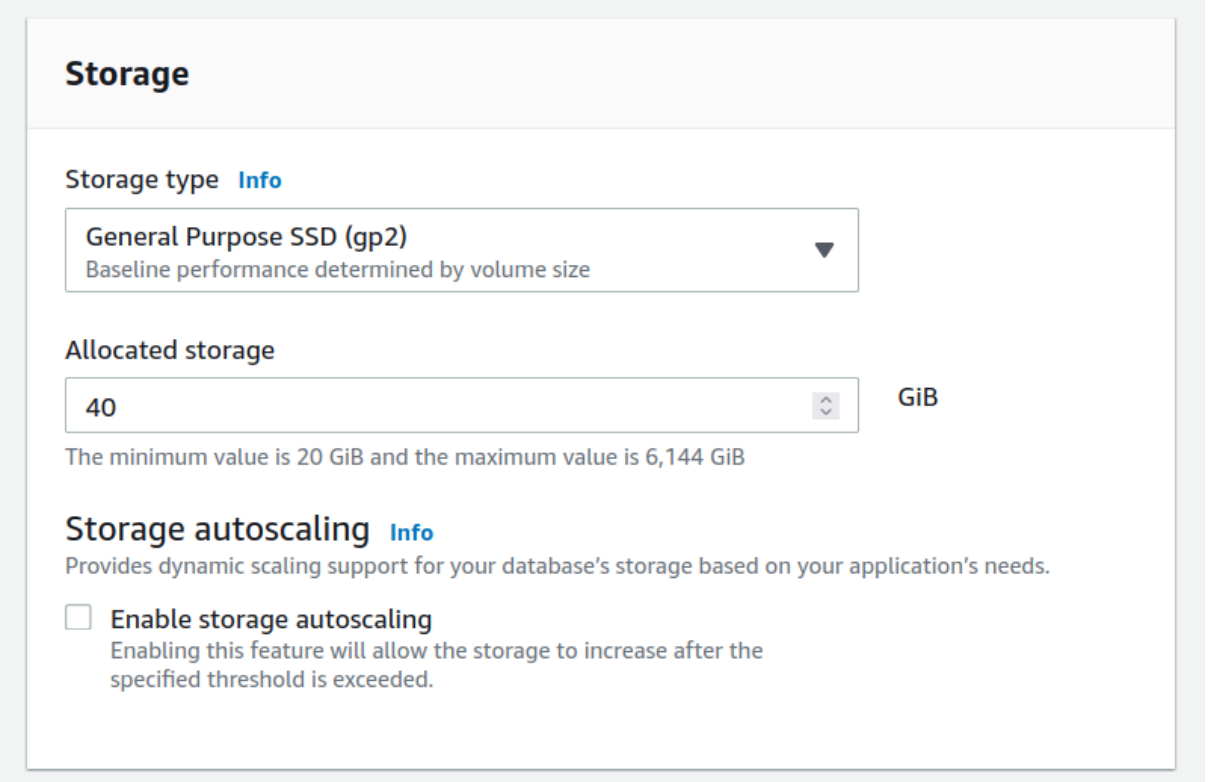
We are using a free-tier template. This is important to avoid accruing costs. We also needed to generate a password in the Credentials Settings.

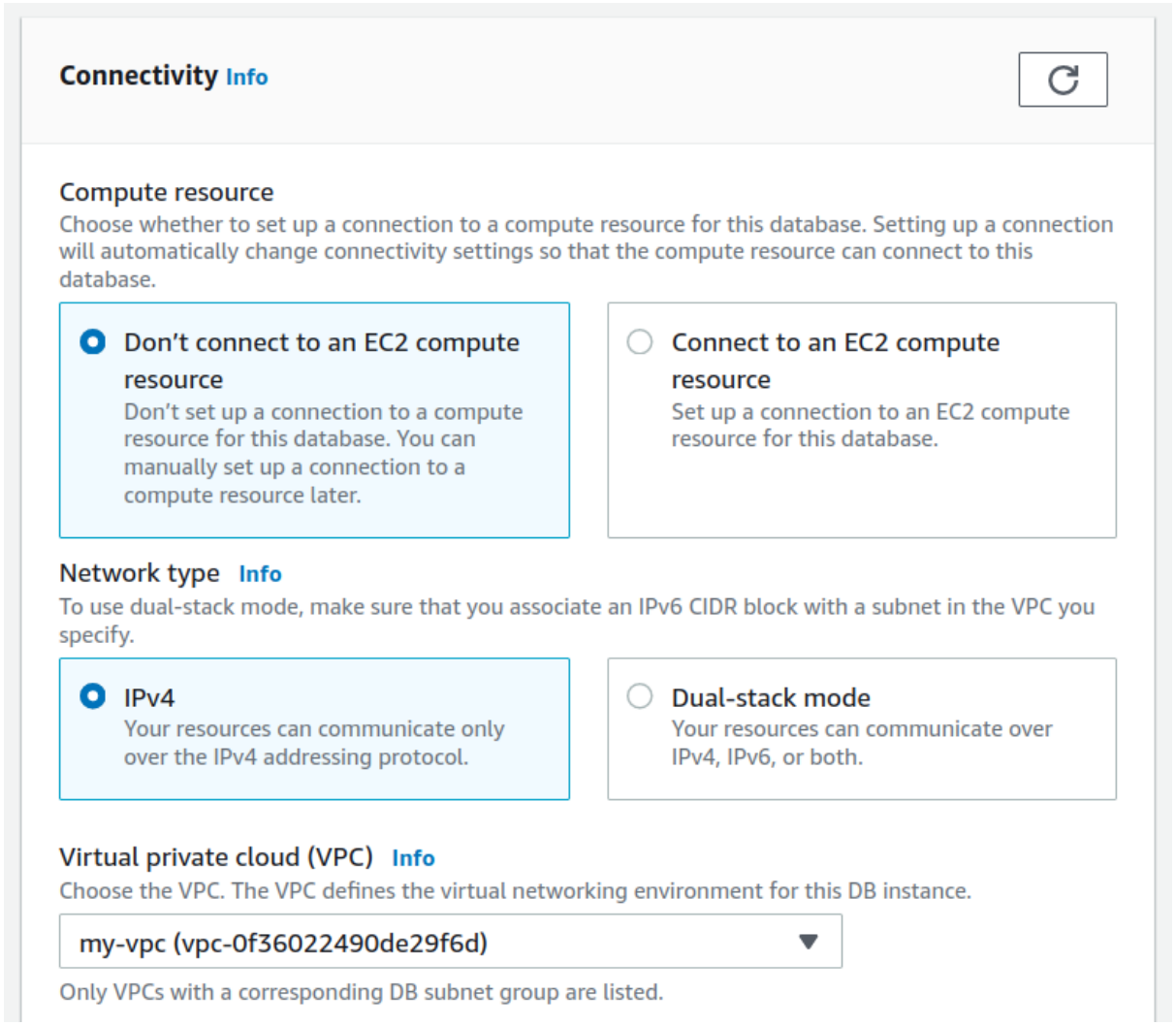




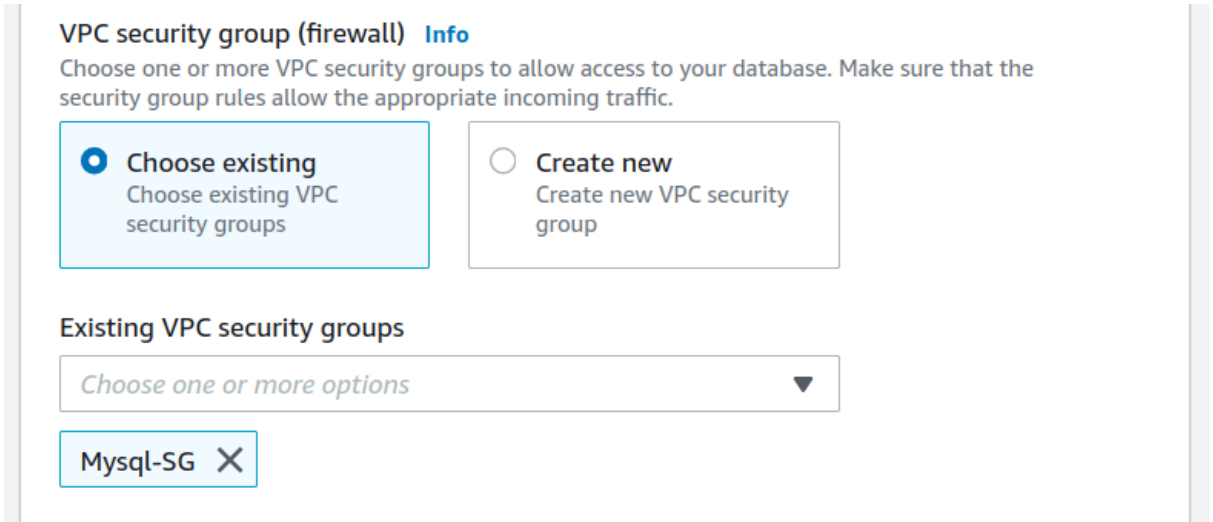
For instance Configuration, We are using db.t2.micro also we uncheck the box labeled “Enable storage autoscaling” .

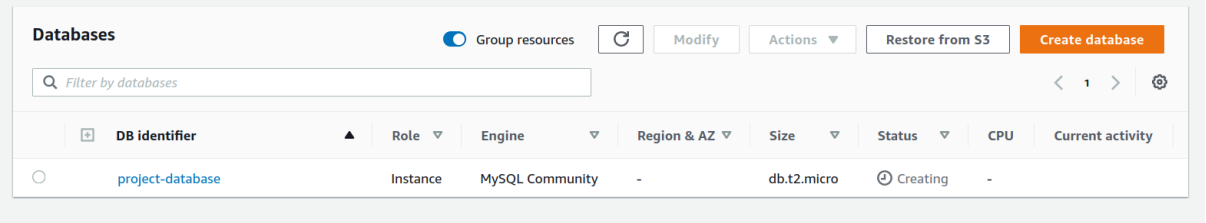




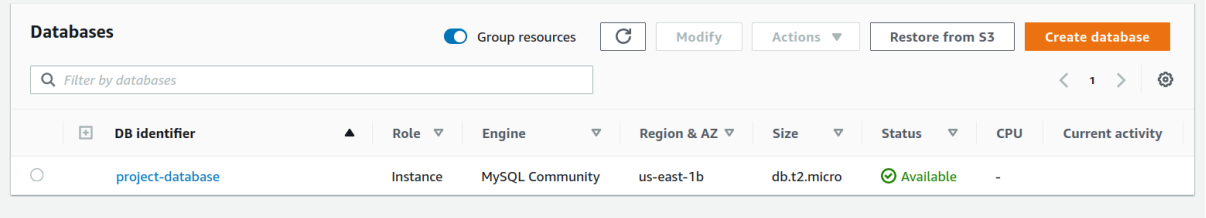


setup the security group for our db instance :

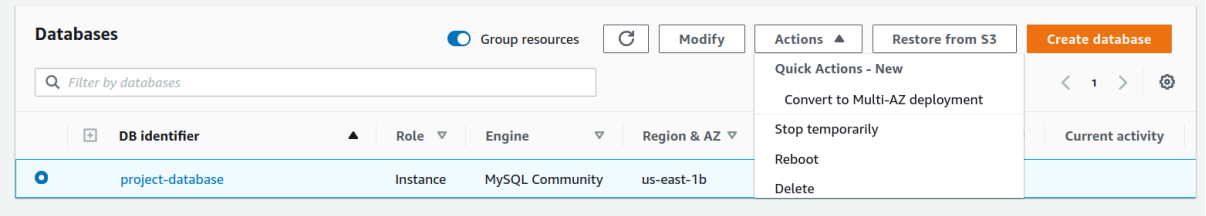


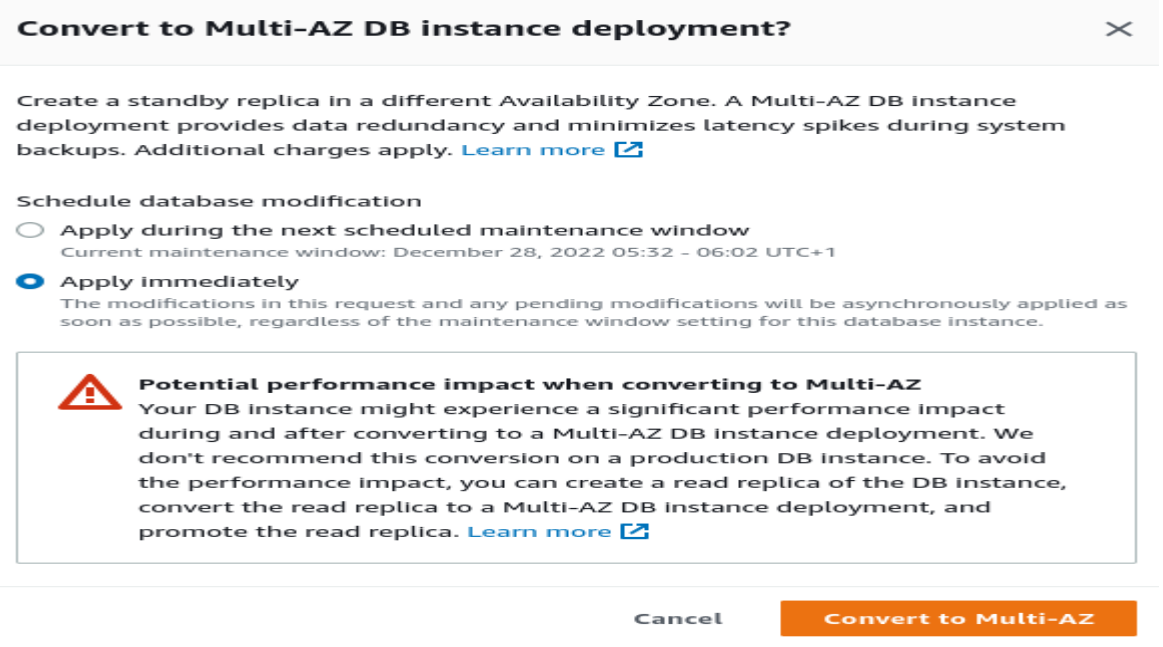


And Now our db instance is Ready as shown below

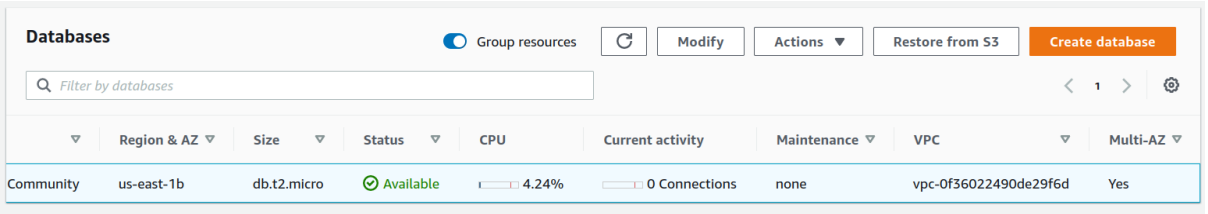


Now let's convert the db instance to Multi-AZ and that will create a standby instance that will be helpful in case of disaster recovery .





And now db-instance become deployed in Multi-AZ :



Connect to our db instance from an Ec2 instance launched in (Wec-App-SG) and have mysql client Already installed on it Then Create a database (crudgames) and table (games)

```
No packages marked for update
[ec2-user@ip-10-0-0-173 ~]$ sudo yum install mysql
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
---> Package mariadb.x86_64 1:5.5.68-1.amzn2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved
```

```
[ec2-user@ip-10-0-0-173 ~]$ sudo mysql -h project-database.chtzczkph8w4.us-east-1.rds.amazonaws.c
om -P 3306 -u oussama -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 231
Server version: 8.0.28 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

```
MySQL [(none)]> create database crudgames;
Query OK, 1 row affected (0.05 sec)
```

```
MySQL [(none)]> show databases
    -> ;
+--------------------+
| Database           |
+--------------------+
| crudgames          |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

Create Table games :

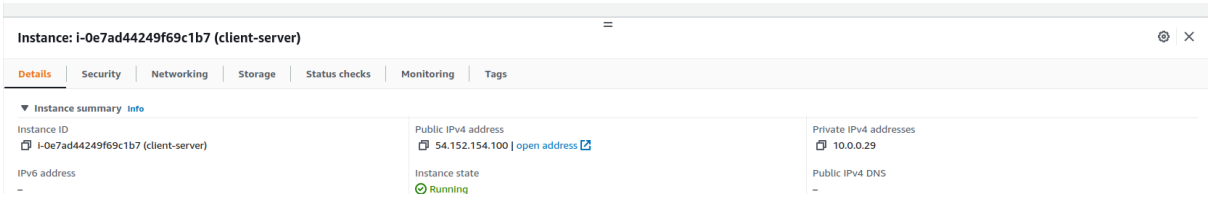
```
MySQL [(none)]> use crudgames
Database changed
MySQL [crudgames]>  create table games(
    -> idgames int not null AUTO_INCREMENT,
    -> name varchar(255),
    -> cost varchar(255),
    -> category varchar(255),
    ->  PRIMARY KEY(idgames))
    -> ;
Query OK, 0 rows affected (0.16 sec)

MySQL [crudgames]>
```

```
MySQL [crudgames]> SHOW TABLES;
+--------------------+
| Tables_in_crudgames |
+--------------------+
| games              |
+--------------------+
1 row in set (0.00 sec)

MySQL [crudgames]>
```

Step 11 : Deploy A React App In EC2 Instance

We deploy the react app in order to interact with our database and perform some crud operations on it .

1 launch an EC2 instance to deploy the React App



then we run that script to pull the project from github repository and run the build

```
#!/bin/bash
# download nvm
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.2/install.sh | bash
# source nvm
chmod +x ~/.nvm/nvm.sh
source ~/.bashrc
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"
# install node
nvm install 16
nvm use 16
#upgrade yum
sudo yum upgrade
#install git
sudo yum install git -y
cd /home/ec2-user
# get source code from githubt
git clone https://github.com/OussamaMaroufi/client-tier.git
#get in server dir
cd client-tier
#give permission
sudo chmod -R 777 .
#install node module
npm install
# start the app
# npm run dev > app1.out.log 2> app1.err.log < /dev/null &
#build the app
npm run build > app1.out.log 2> app1.err.log < /dev/null &
# to redirect traffic comes to port 80 into port 3001
sudo iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 3000
```

we run this bash script to install nginx server

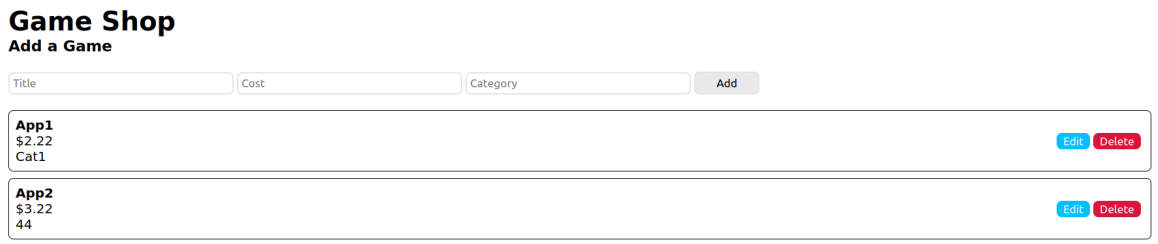
```
#!/bin/bash
sudo yum update -y
sudo amazon-linux-extras install nginx1 -y
sudo systemctl enable nginx
sudo systemctl start nginx
```

add this configuration in /etc/nginx/conf.d

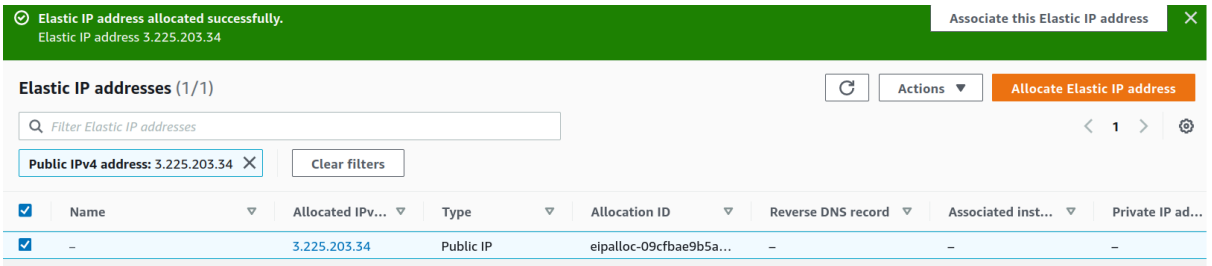
```
server {
  listen 80;
  listen [::]:80;
  root /home/ec2-user/app-deploy/dist;
  location / {
   try_files $uri /index.html;

  }
}
```

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[ec2-user@ip-10-0-0-29 conf.d]$ sudo service nginx reload
Redirecting to /bin/systemctl reload nginx.service
[ec2-user@ip-10-0-0-29 conf.d]$
```

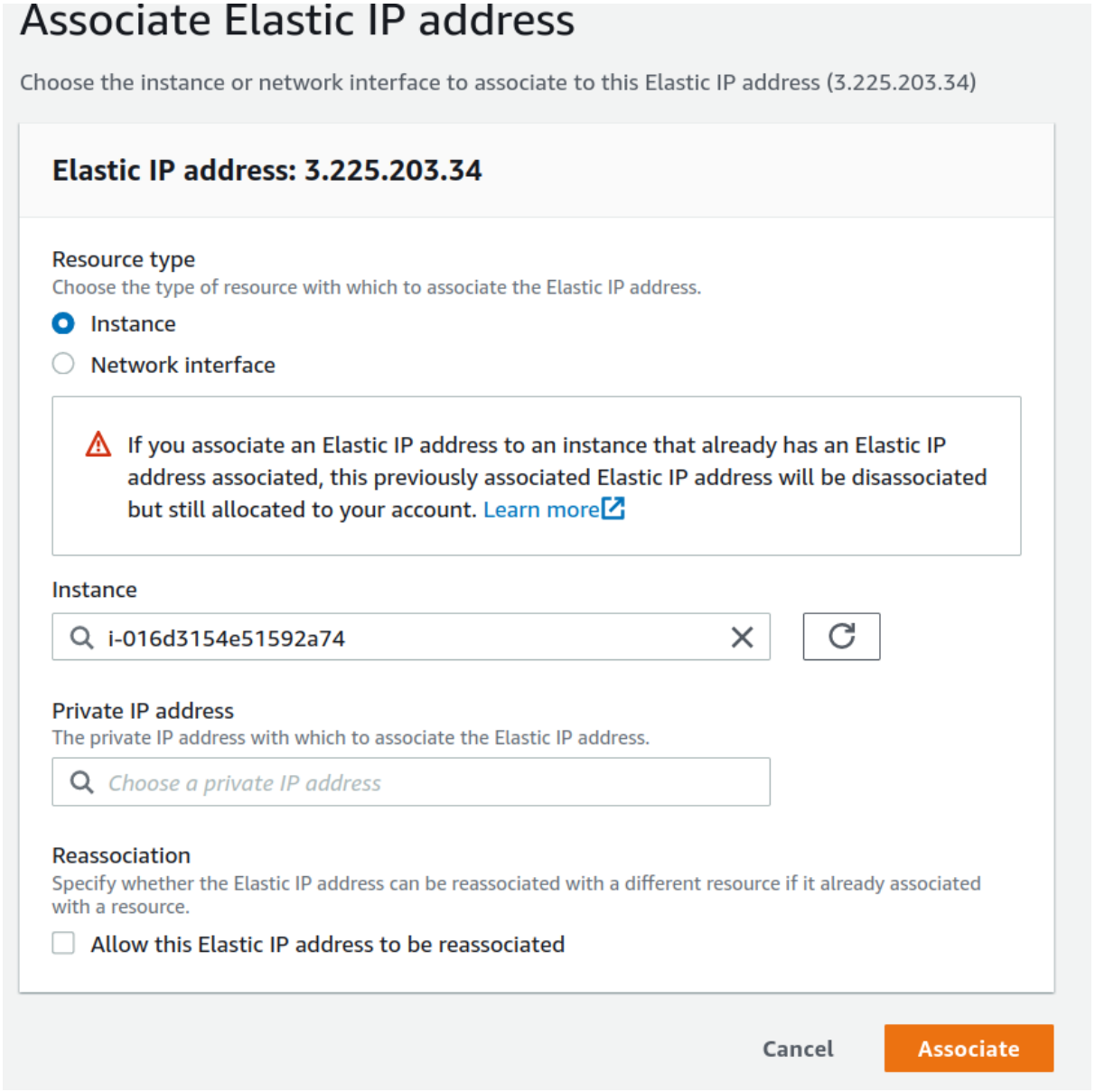
our app is deployed successfully



Now we should create an Elastic Ip and attach it to our client-server



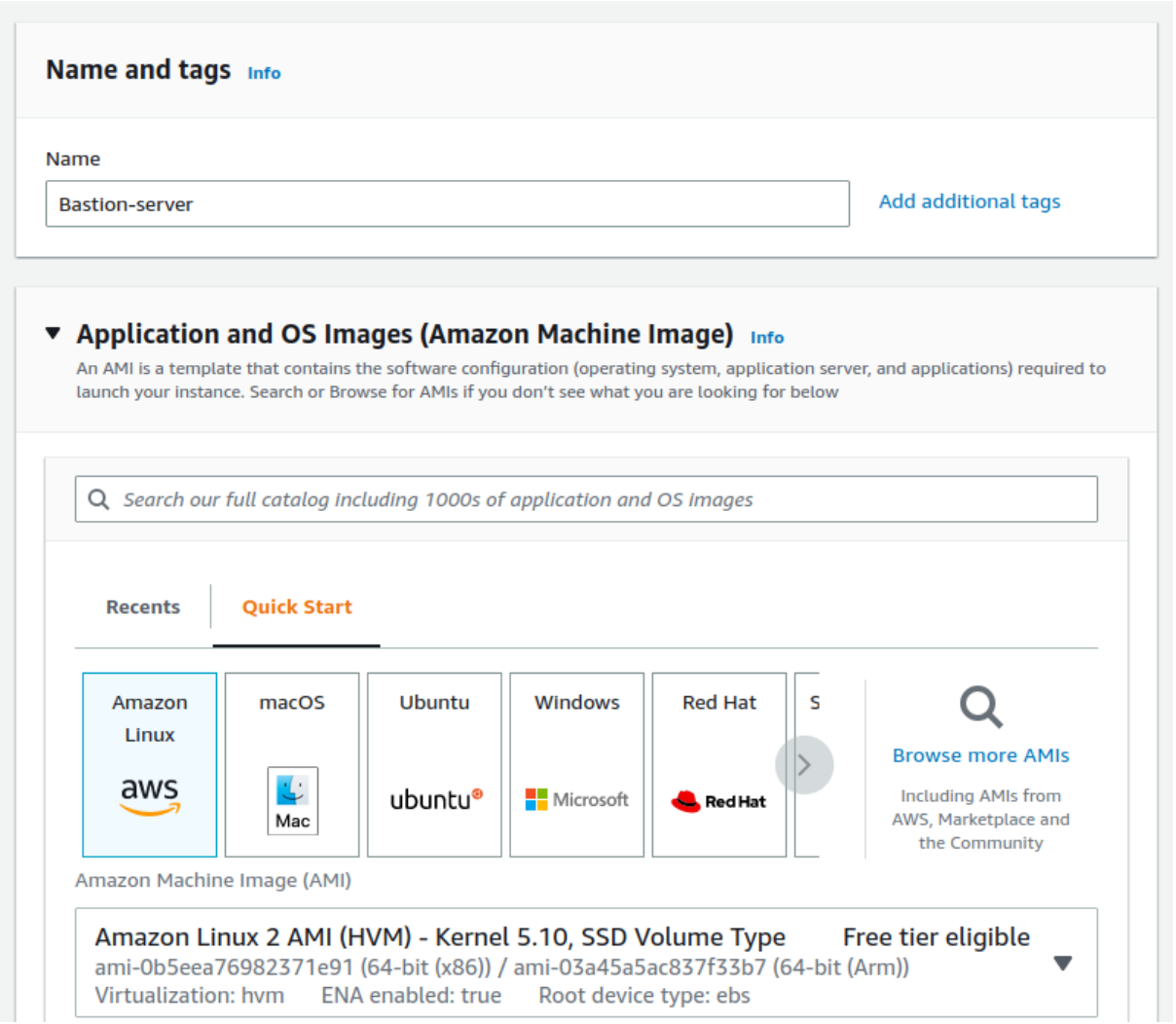
After we allocate an Elastic Ip we Need to associate it to the instance running our frontend tier .

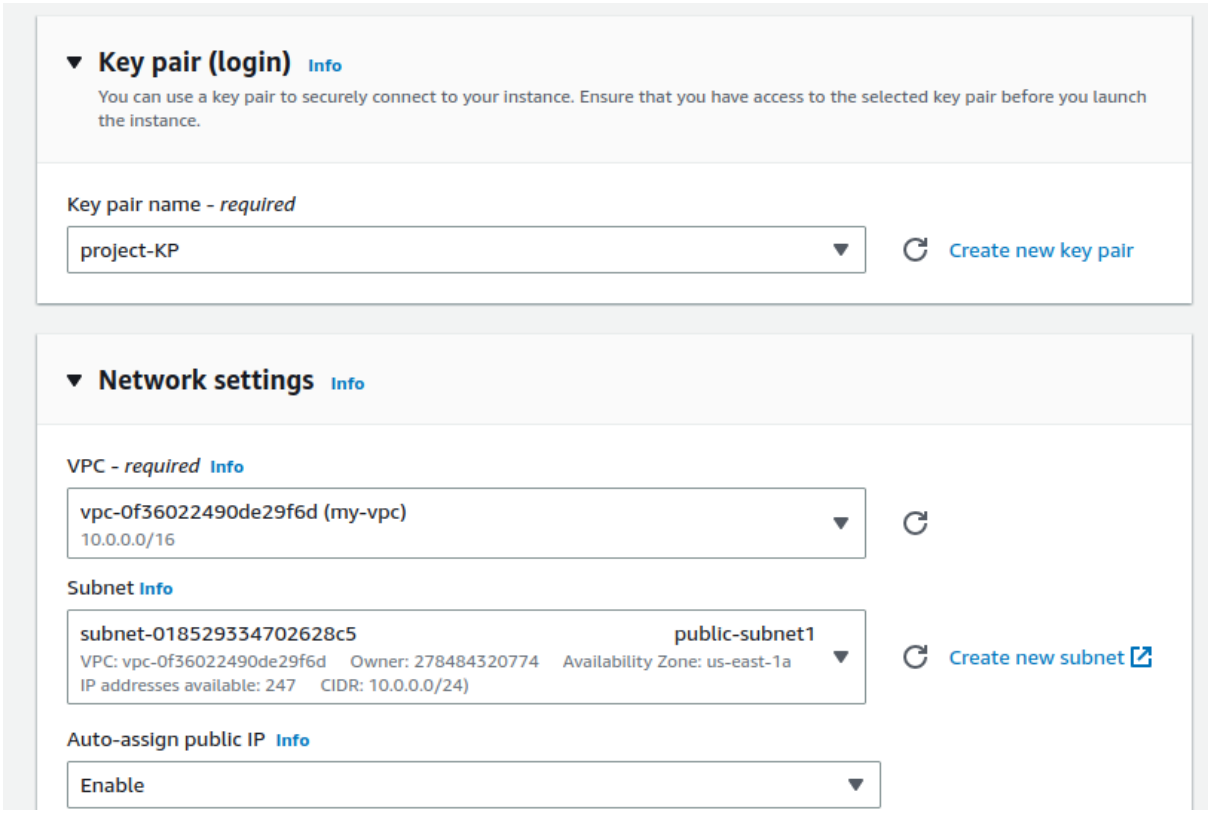


and now our application is available on http://3.225.203.34

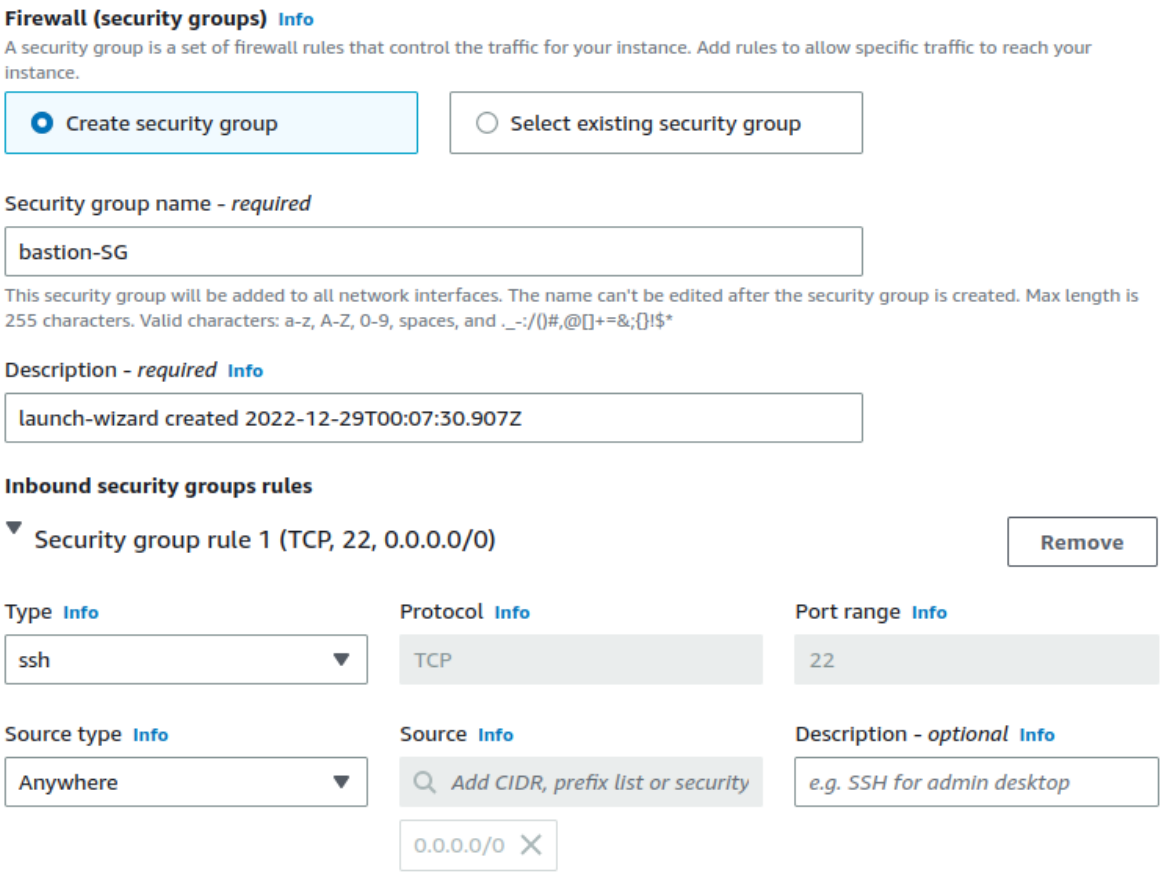
Step 12 : Create Machine Bastion

1 - create a security group for bastion machine

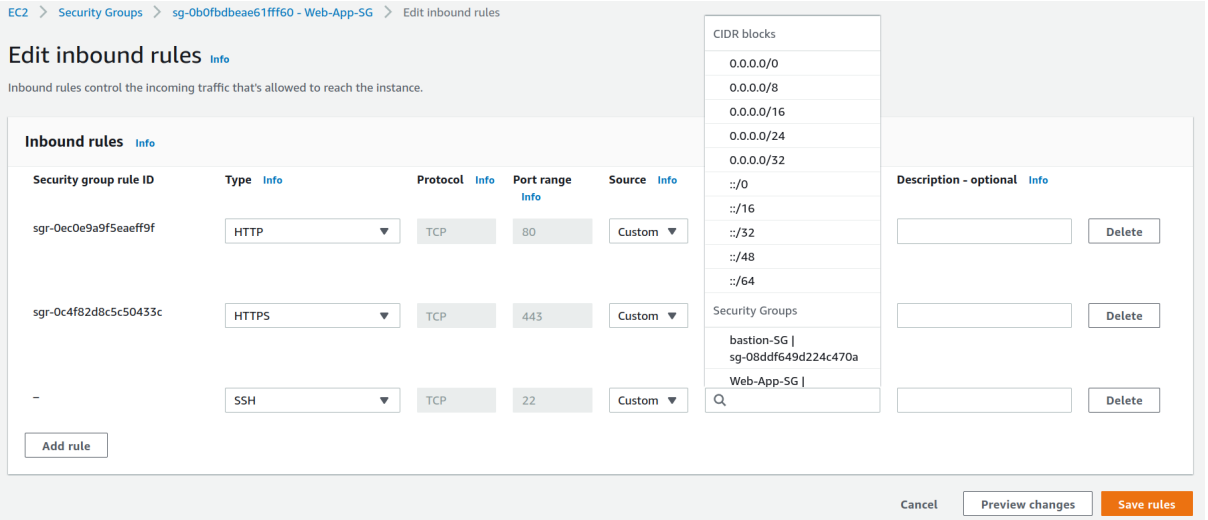




We allow ssh access from everywhere to the bastion machine .



2 - Edit Web-App-SG to allow ssh traffic from bastion-SG



III. Conclusion

OK that's it! After setting up a VPC, subnets, route table, security group, and internet gateway,nat gateway,, you then created a Launch Template with specific AMI, instance type, key pair, security group, and user data. We then created an Auto Scale group using the Launch Template, and applied an Application Load Balancer. Our business application is now ready to meet the demands of its users .

Références :

- **https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html**
- https://docs.aws.amazon.com/autoscaling/ec2/userguide/auto-scaling-groups.html
- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html
- https://docs.amazonaws.cn/en_us/AmazonRDS/latest/UserGuide/Welcome.html
- https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html
- https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html