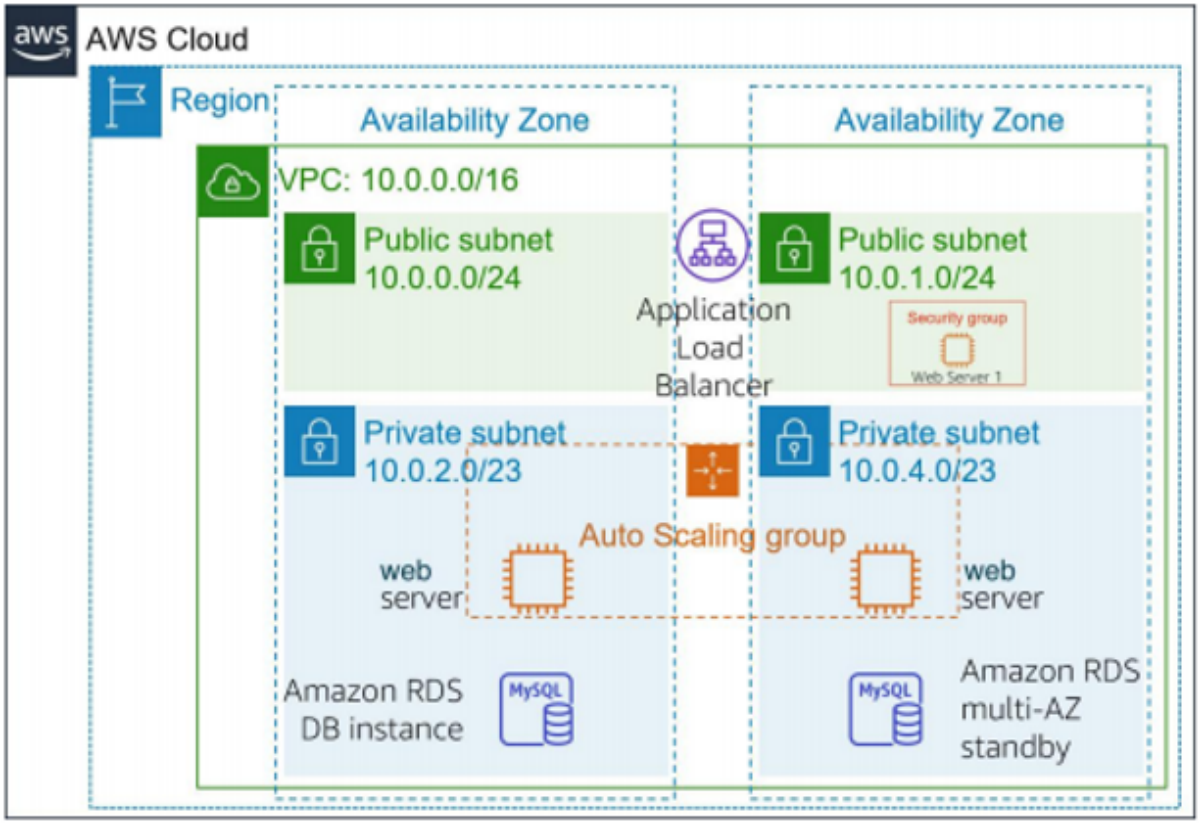
Livrable du projet AWS

Déploiement d'une application web hautement disponible (HA) sur AWS



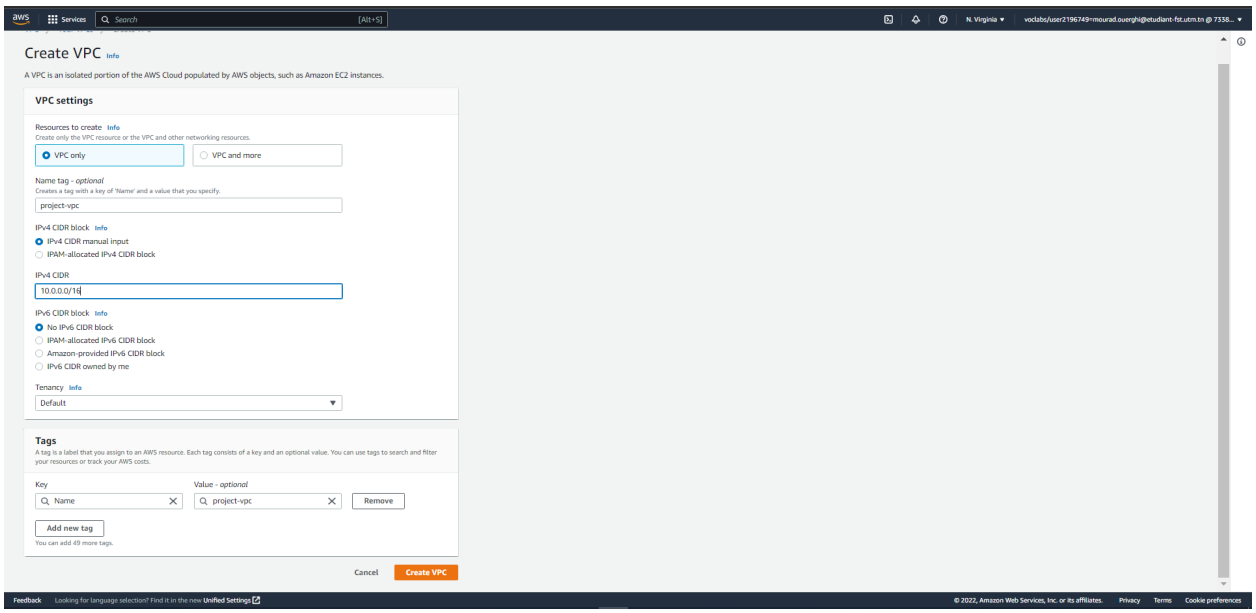
Ouerghi Mourad

IGL 4

Table de matières

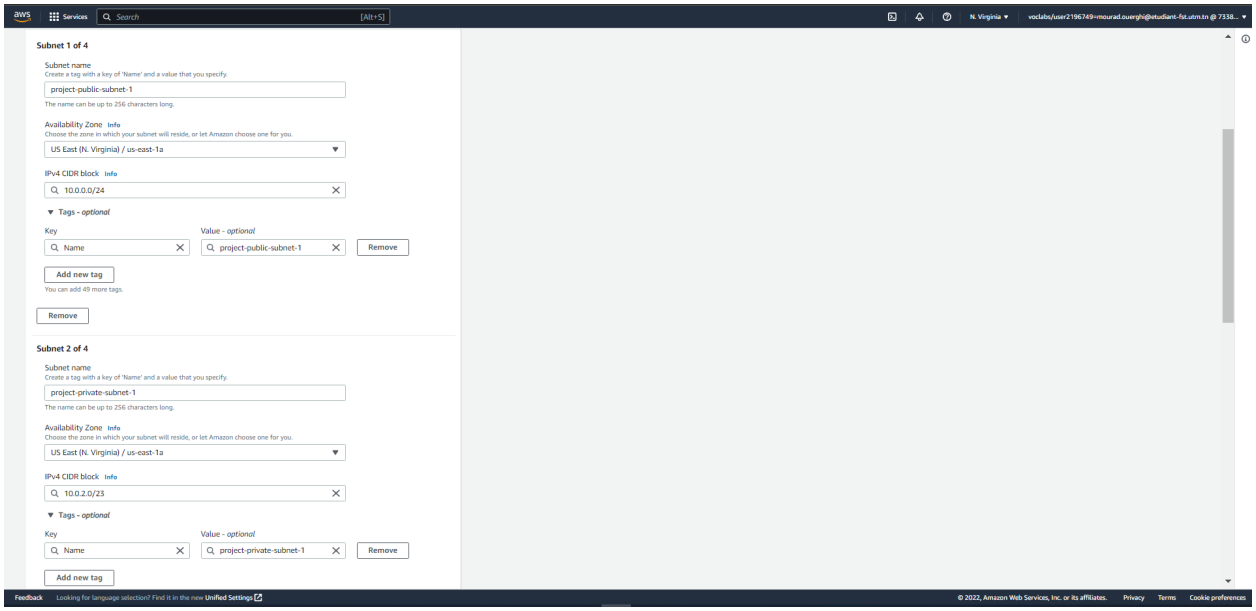
1. Création du VPC

Dans cette partie j’ai créé un vpc avec un IPV4 CIDR = 10.0.0.0/16 sous le nom de “project-vpc”

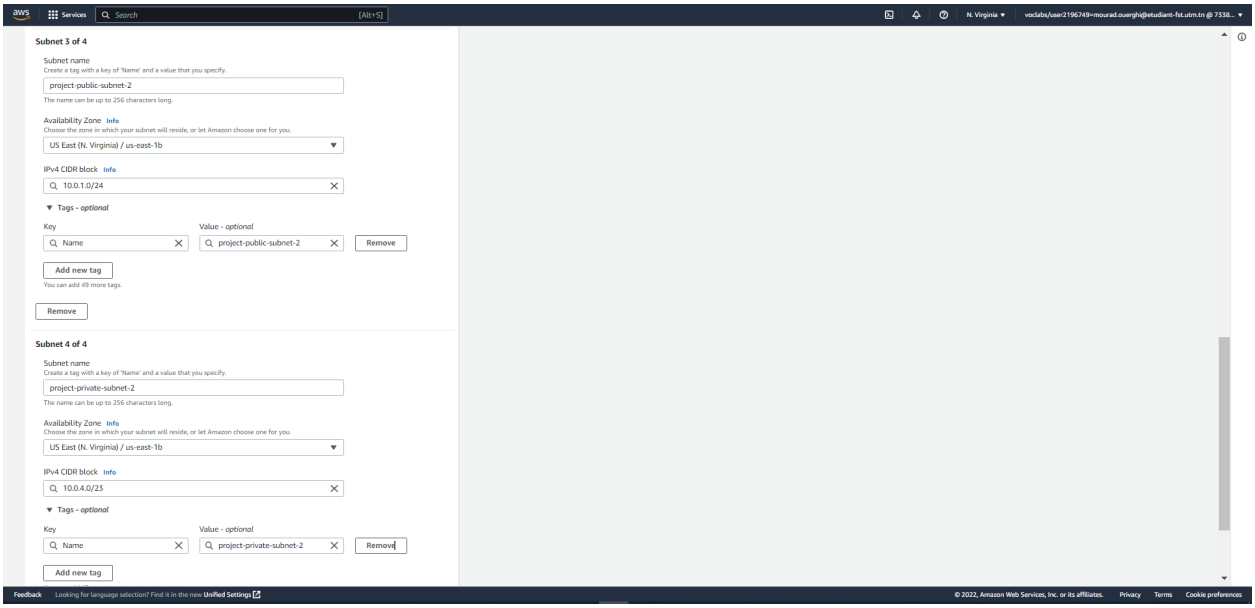


2. Création des subnets

Dans cette étape j’ai créé une public subnet sous le nom de project-public-subnet 1 avec un CIDR IPV4 = 10.0.0.0/24 et une private subnet le nom de project-private-subnet 1 avec un CIDR IPV4 = 10.0.0.2/23, les deux dans l’availability zone us-east-1a .



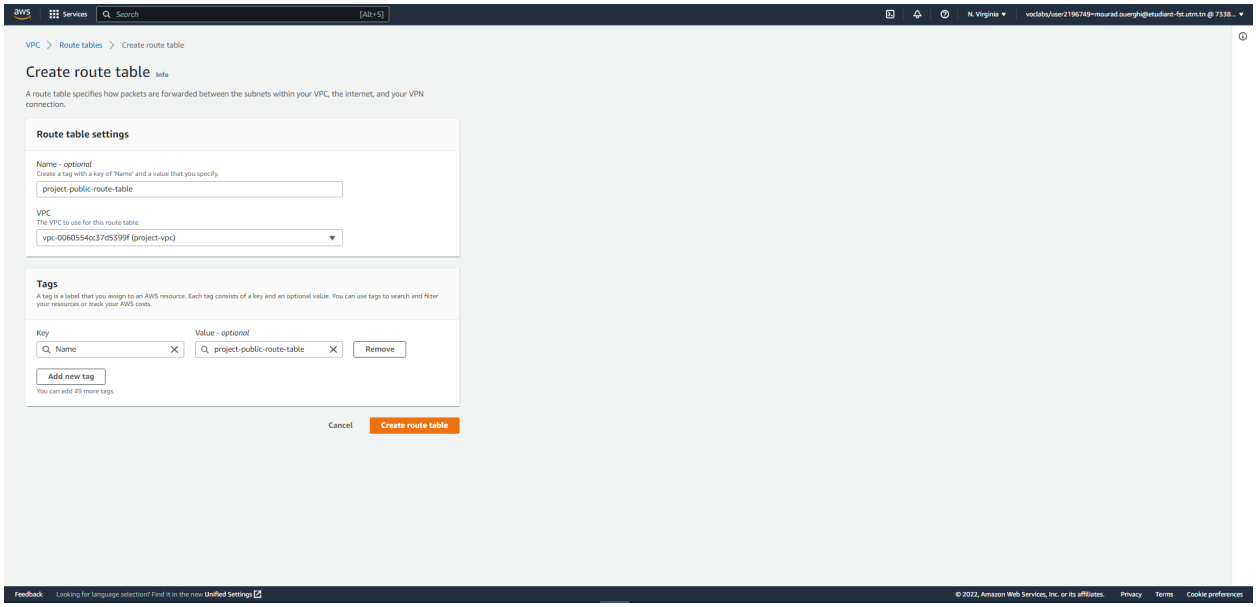
Dans cette étape j’ai créé une public subnet sous le nom de project-public-subnet 2 avec un CIDR IPV4 = 10.0.1.0/24 et une private subnet le nom de project-private-subnet 2 avec un CIDR IPV4 = 10.0.4.0/23, les deux dans l’availability zone us-east-1b .



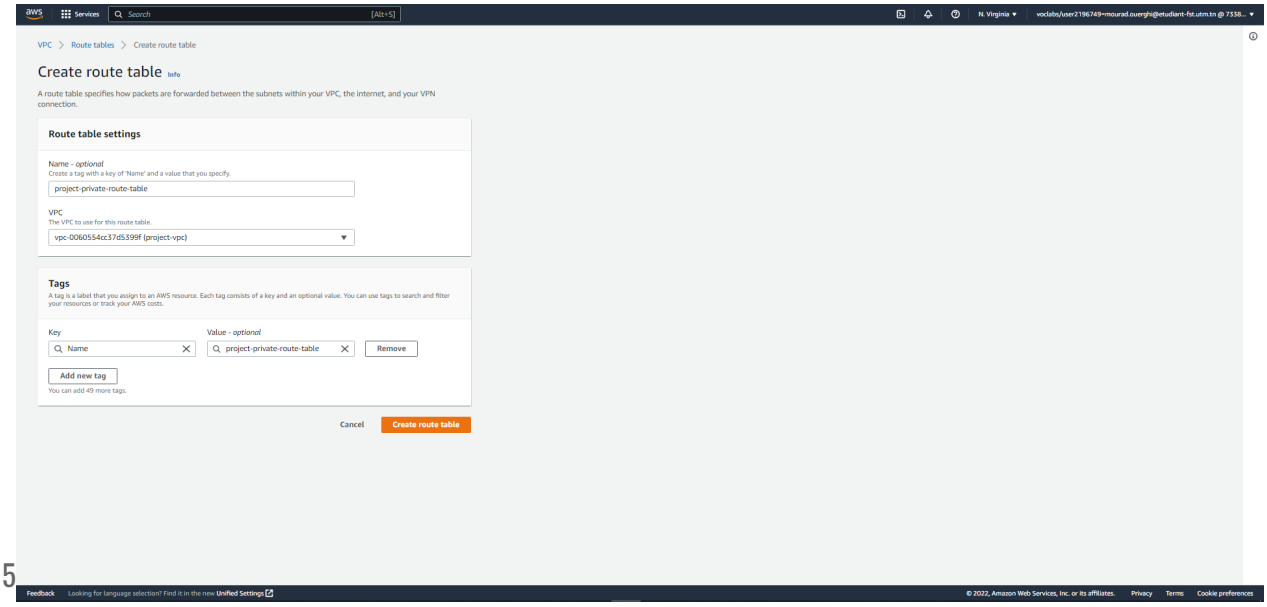
3. Routes tables

a. Création des route tables

Dans cette étape j’ai créé une public route table sous le nom de project-public-route-table dans le vpc “project-vpc”

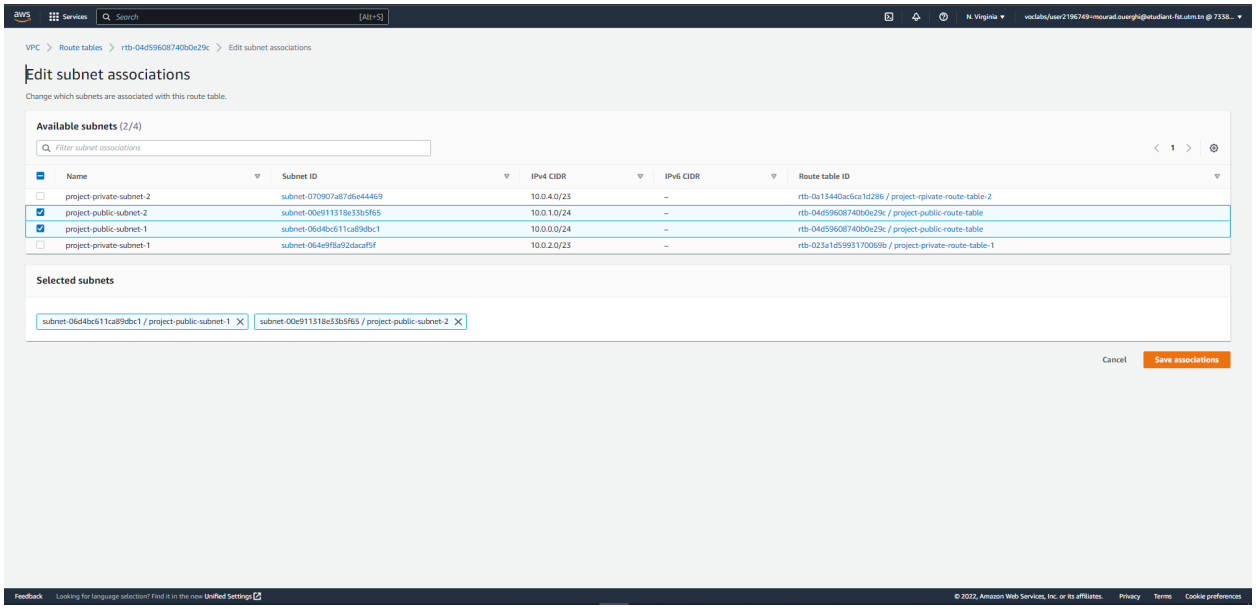


Dans cette étape j’ai créé une private route table sous le nom de project-private route table dans le vpc “project-vpc” et j’ai créé une deuxième ensuite puisque on utilisera deux nat gateways ensuite

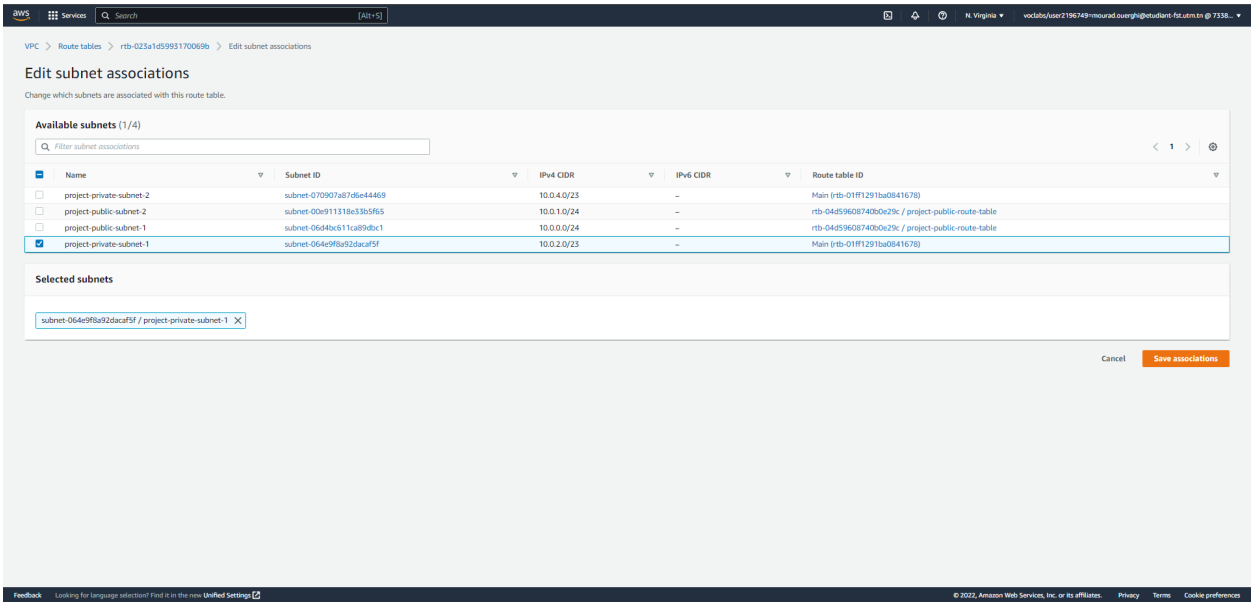


b. Association des route tables

Association des deux public subnets créé précédemment avec public route table

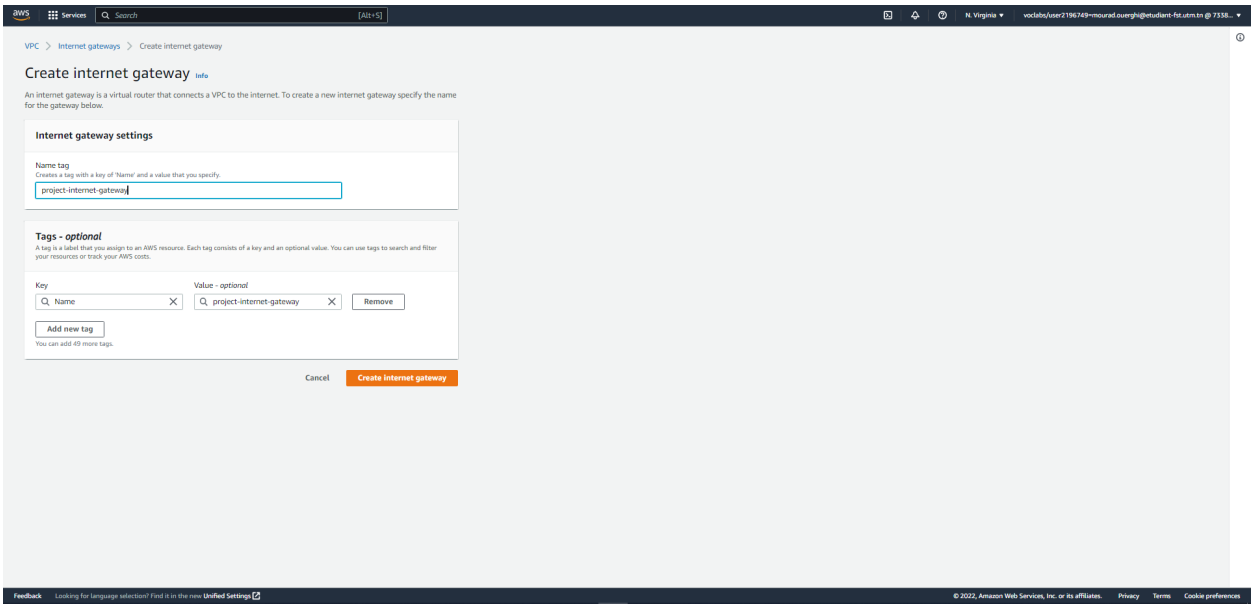


Association du private subnet “project-private-subnet-1” avec le “project-private-roue-table” et j’ai refait la même étape avec le “project-private-subnet-2” mais avec le “project-private-roue-table-2”

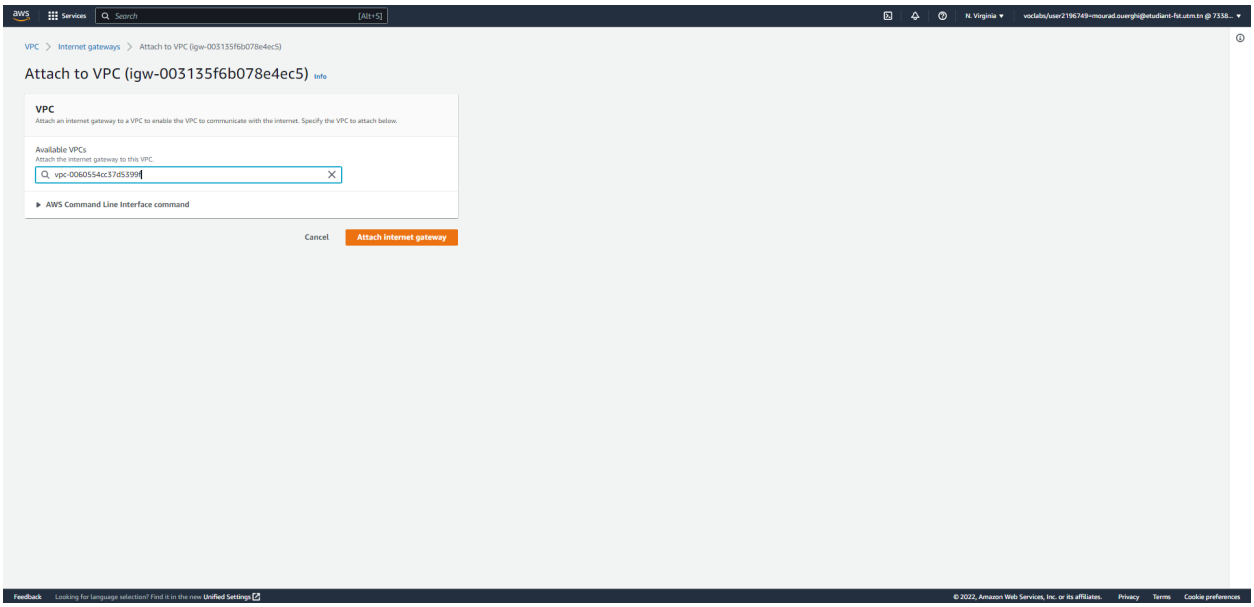


4. Création du Internet Gateway

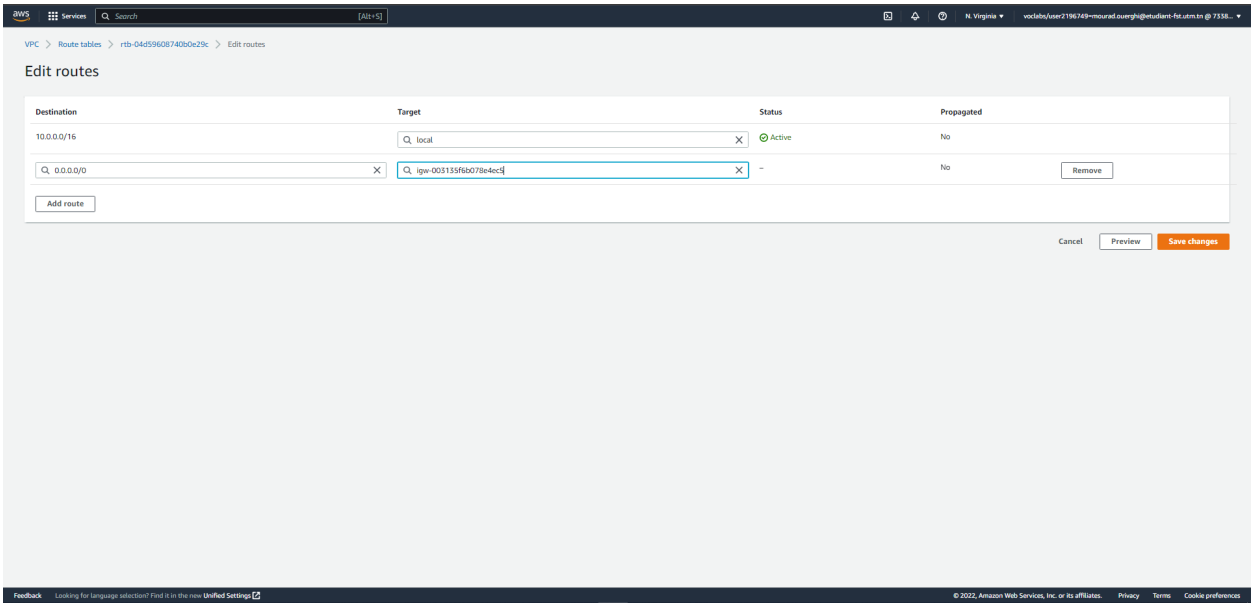
Création d’un Internet Gateway sous le nom de “project-internet-gateway”



Liaison du internet Gateway créé avec le “project-vpc”

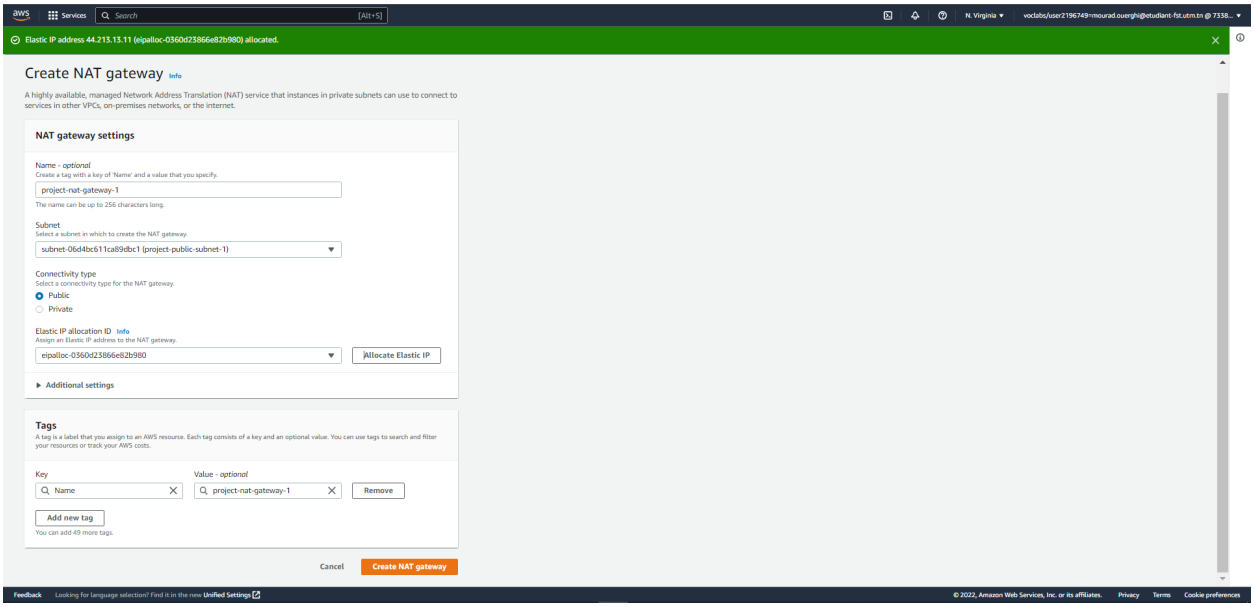


Ajout d’une route vers le internet gateway créé dans le public route table

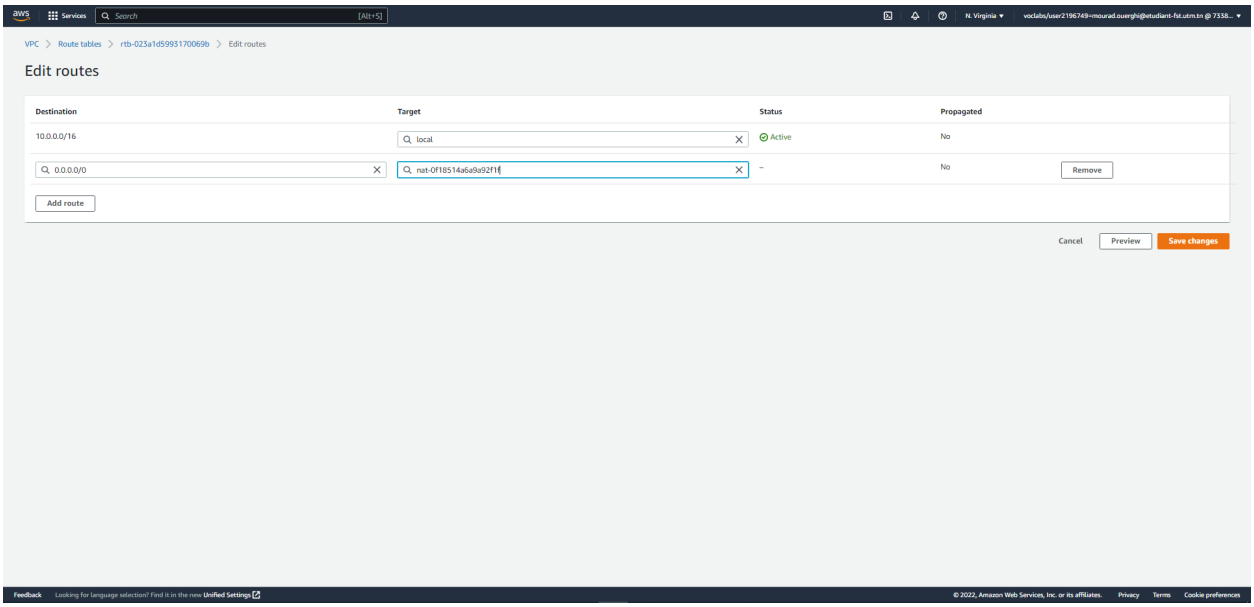


5. Création des Nat-Gateways

Création d’un nat gateway sous le nom de “project-nat-gateway-1” dans le “public-subnet-1”. J’ai refait la même étape avec un “project-nat-gateway-2” dans le “public-subnet-2”



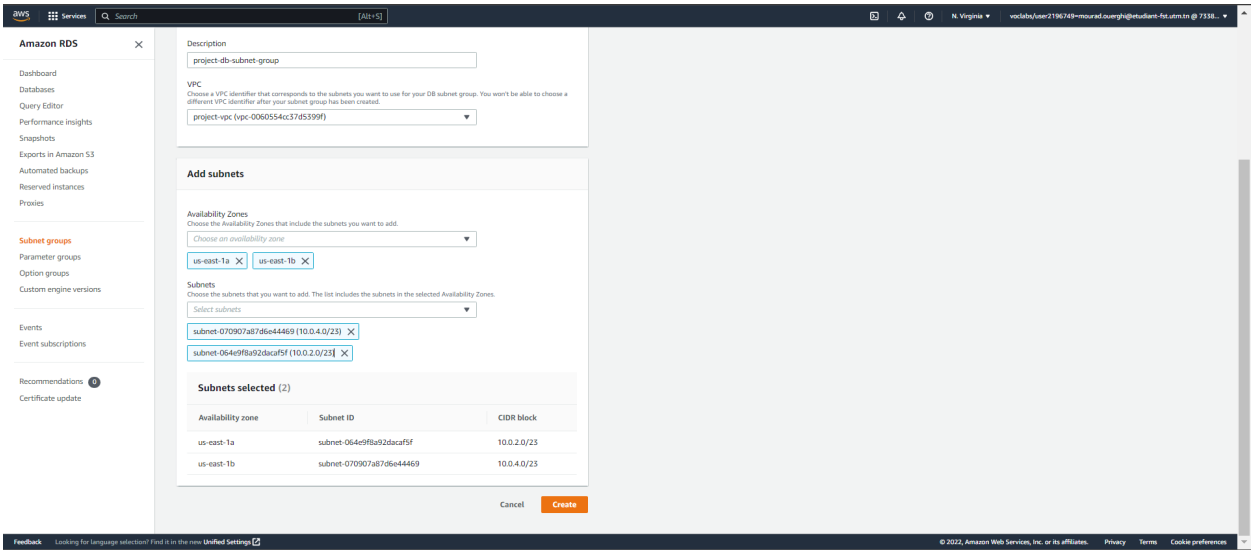
Ajout d’une route vers le nat gateway créé dans le private route table et j’ai refait la même étape avec le deuxième nat gateway et le deuxième route table



6. Création d’une base de données MYSQL avec RDS

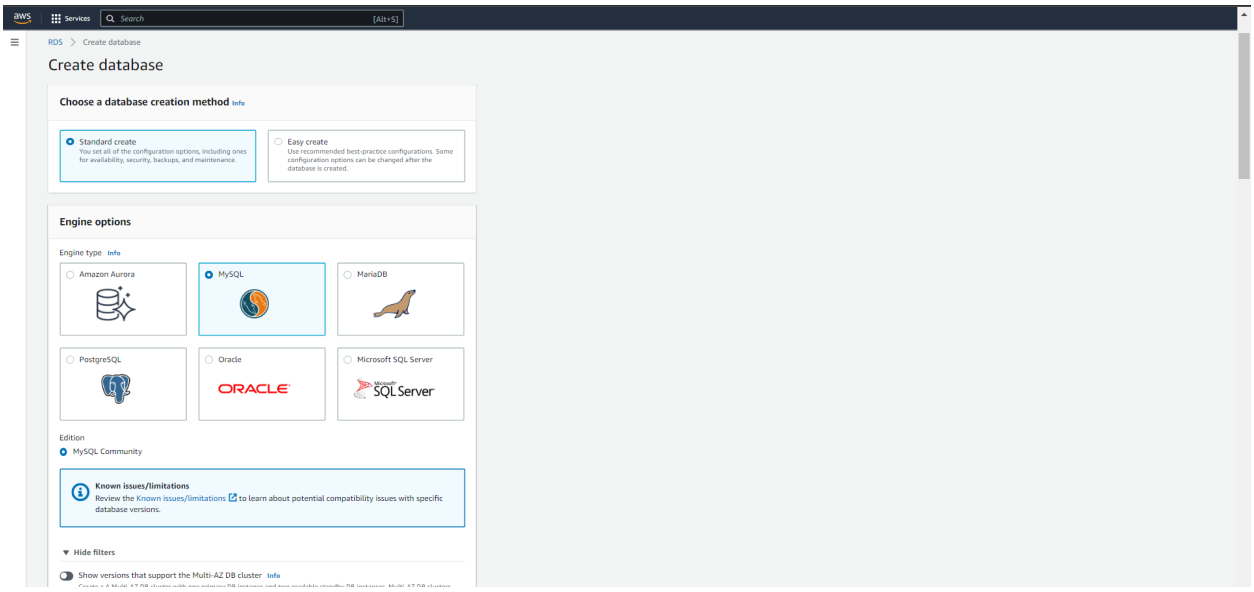
a. Subnet group

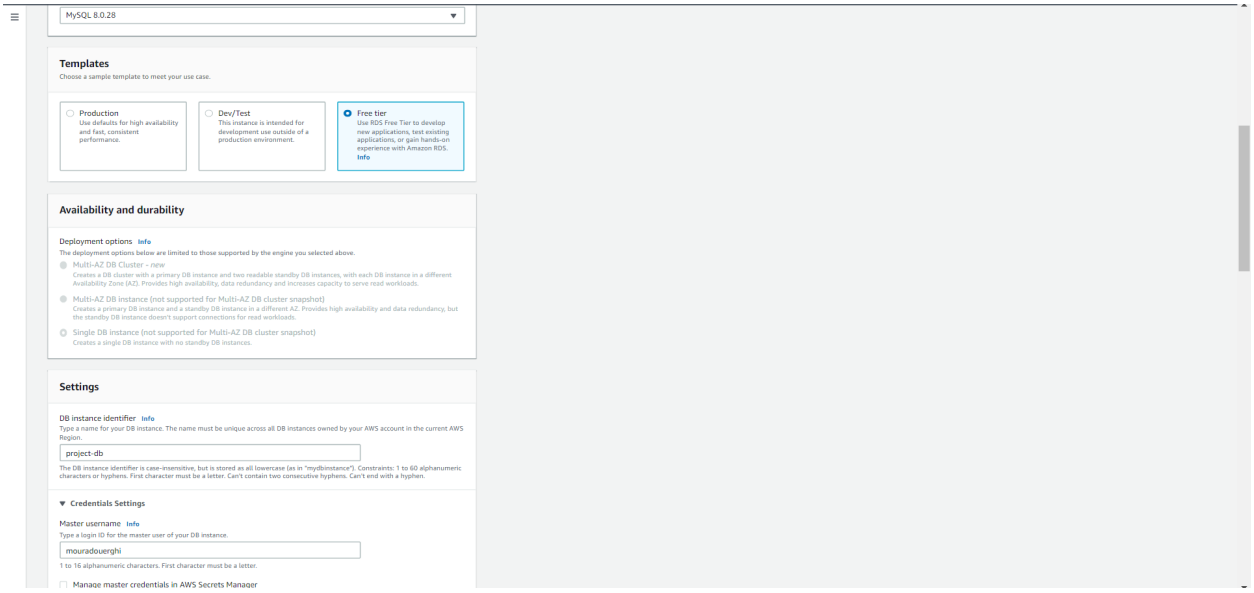
Création d’un subnet group qui avec les deux private subnet groups pour l’utiliser ensuite dans la création d’une instance RDS



b. L’instance RDS

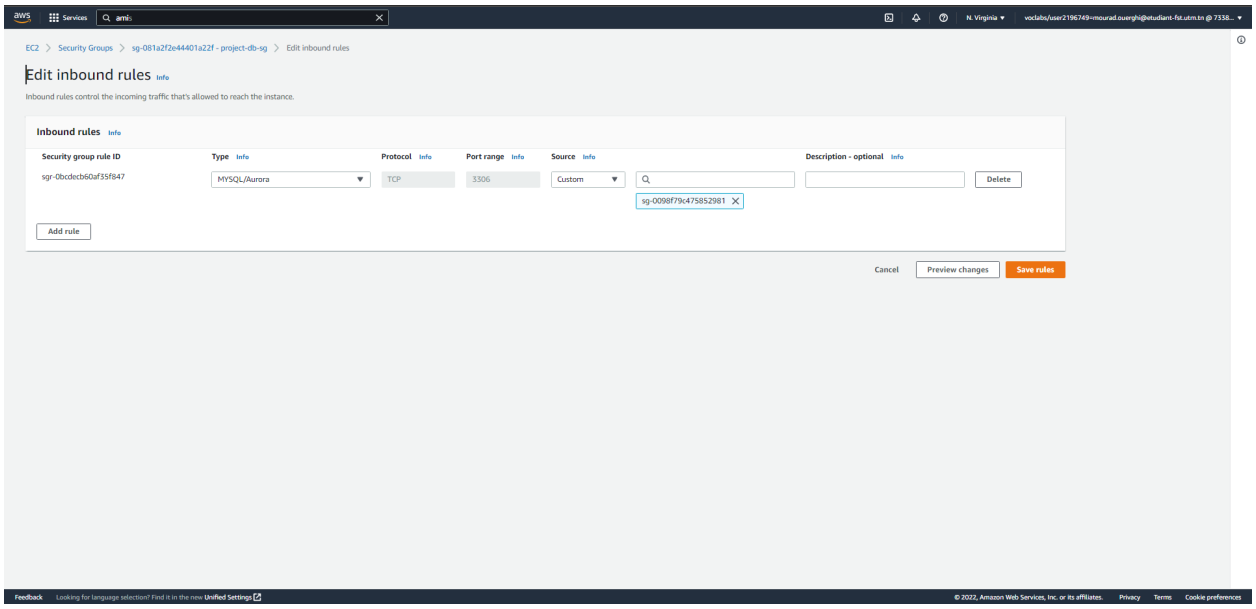
Création d’une instance RDS avec Engine MYSQL sous le nom de “project-db”





c. Security Group

C’est le security group du "projet-db" après avoir terminé le projet, contenant une seule inbound rule qui permet aux instances du web server d’accéder à la base de données rds

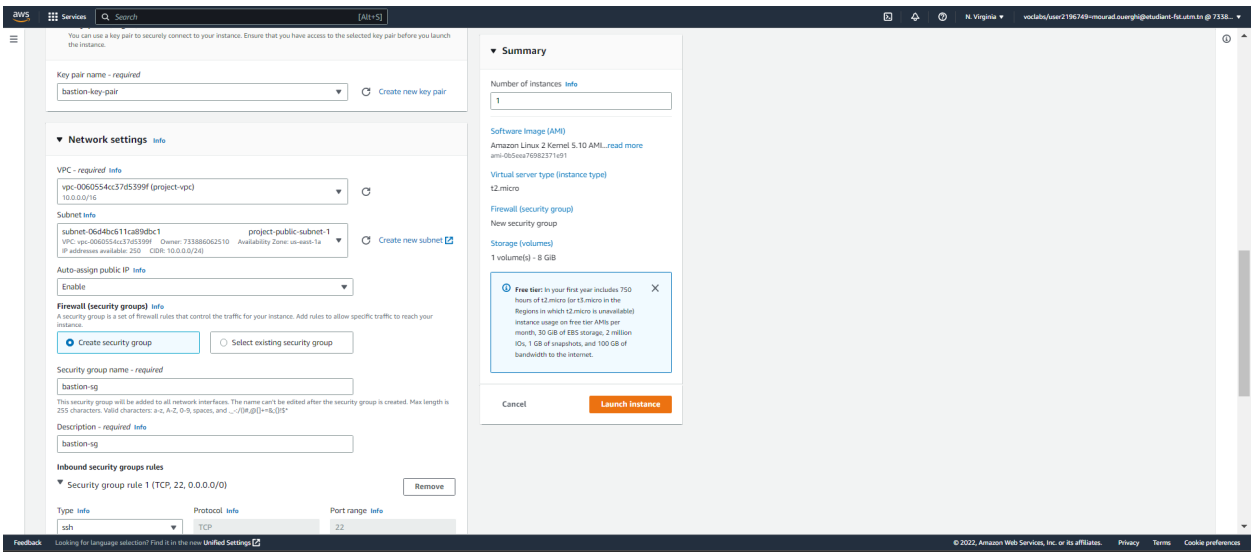


7. Création des instances EC2

a. Création du machine bastion

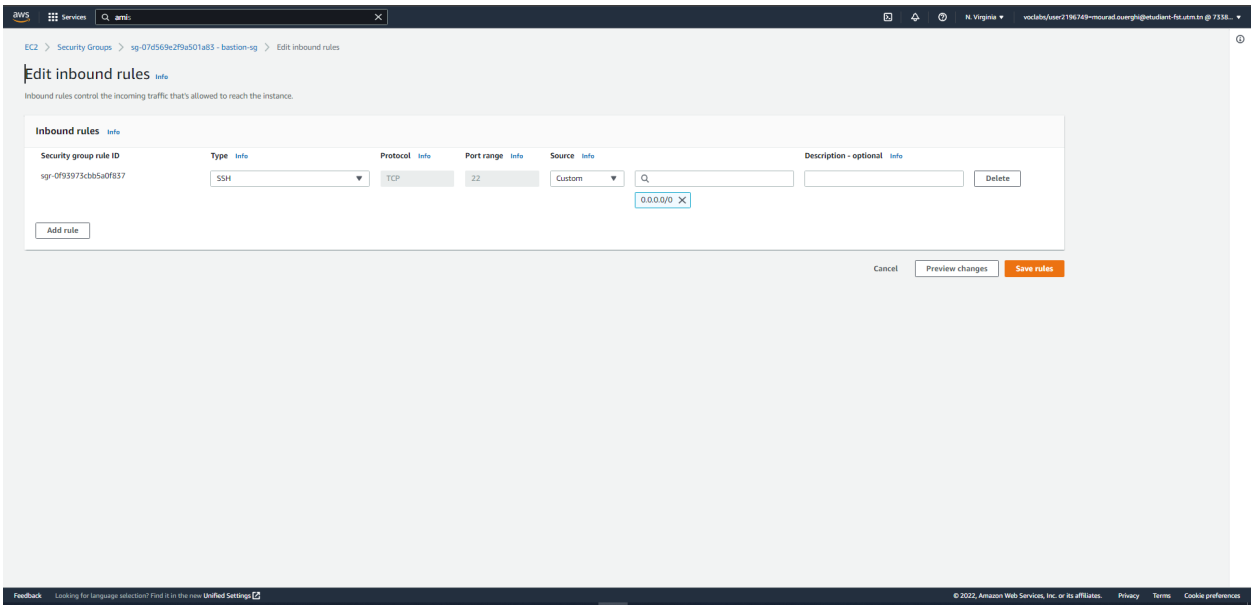
i. L’instance Bastion

Création d’une machine bastion basé sur Amazon Linux de type t2.micro dans le public subnet “project-public-subnet-1” avec une clé ssh que j’ai téléchargé pour pouvoir accéder à la machine ensuite



ii. Security Group

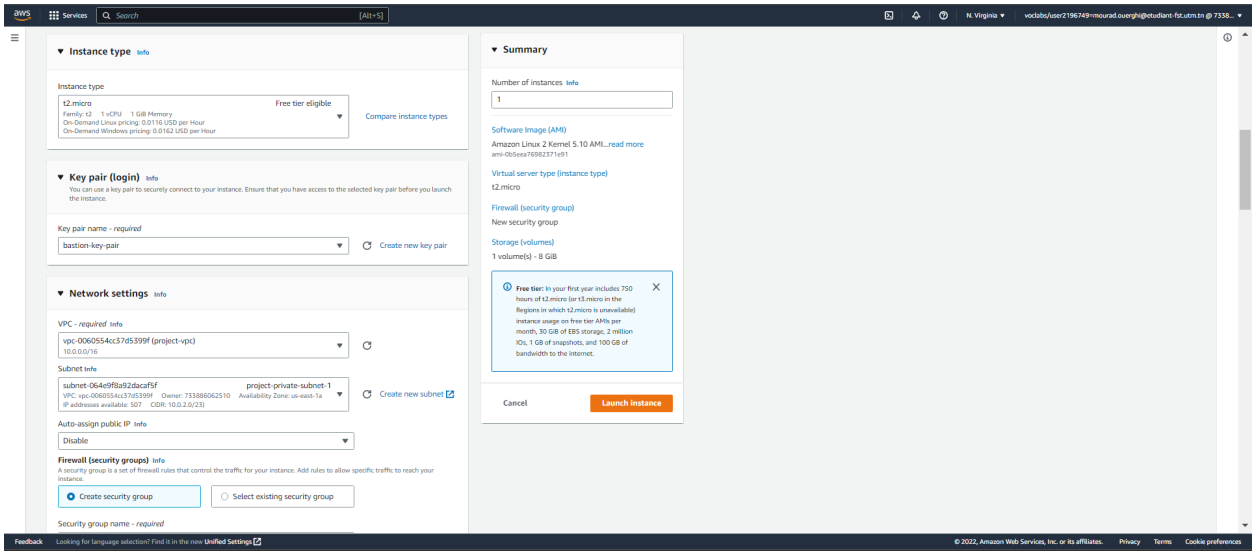
Le security group du machine bastion ne permet l'accès qu'à travers ssh de n’importe quelle source

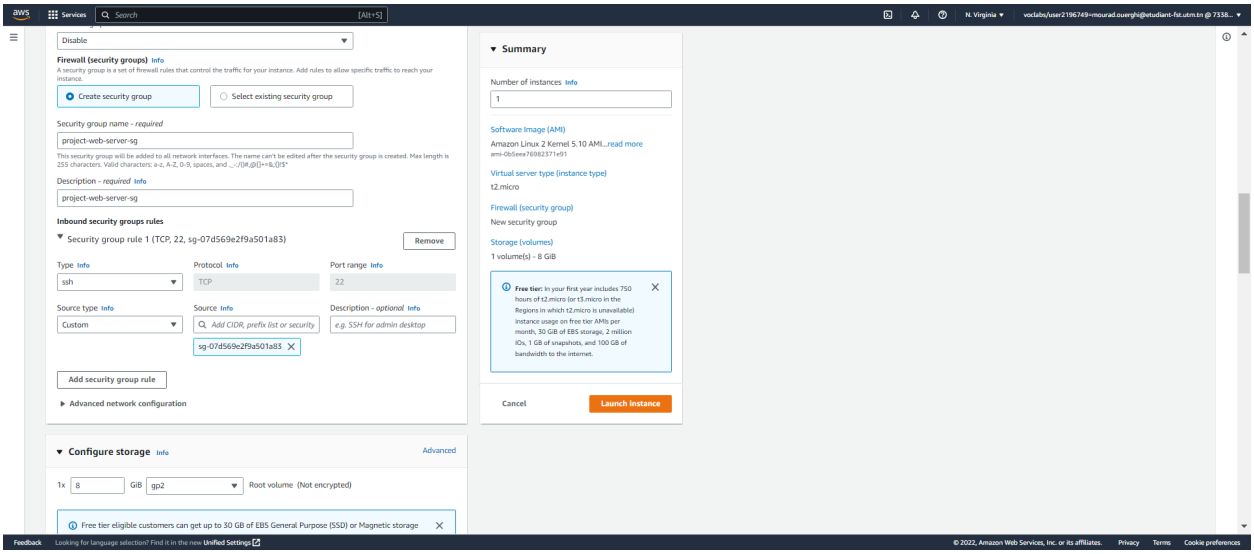


b. Création du machine Web Server

i. L’instance Web Server

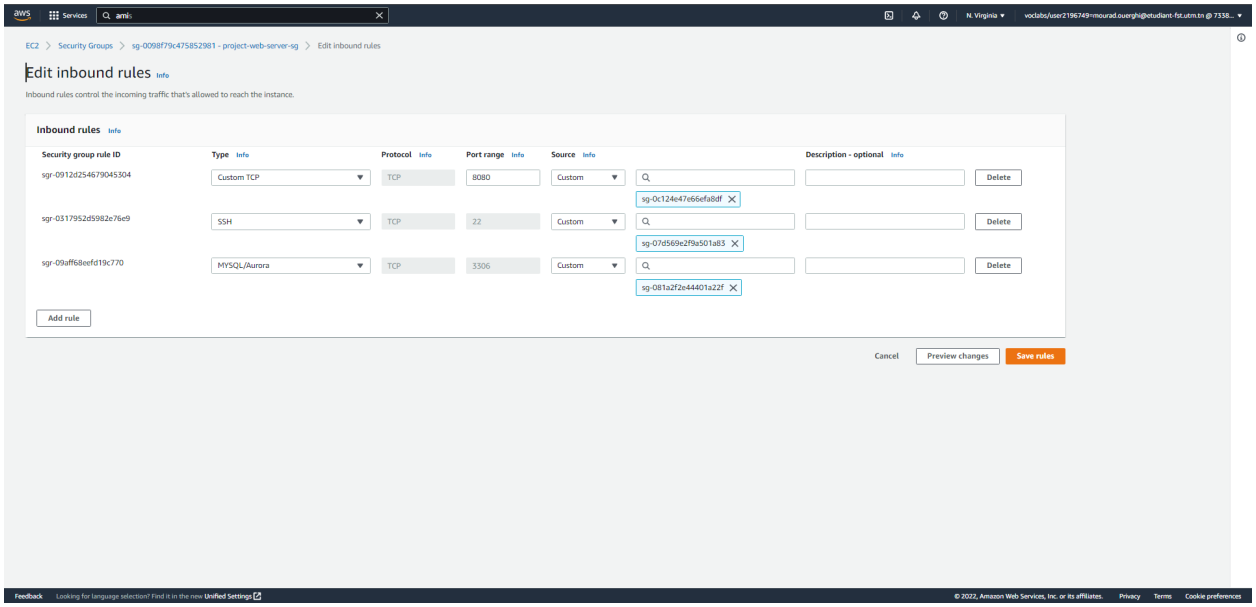
C’est presque les même configurations du machine bastion mais dans une private subnet





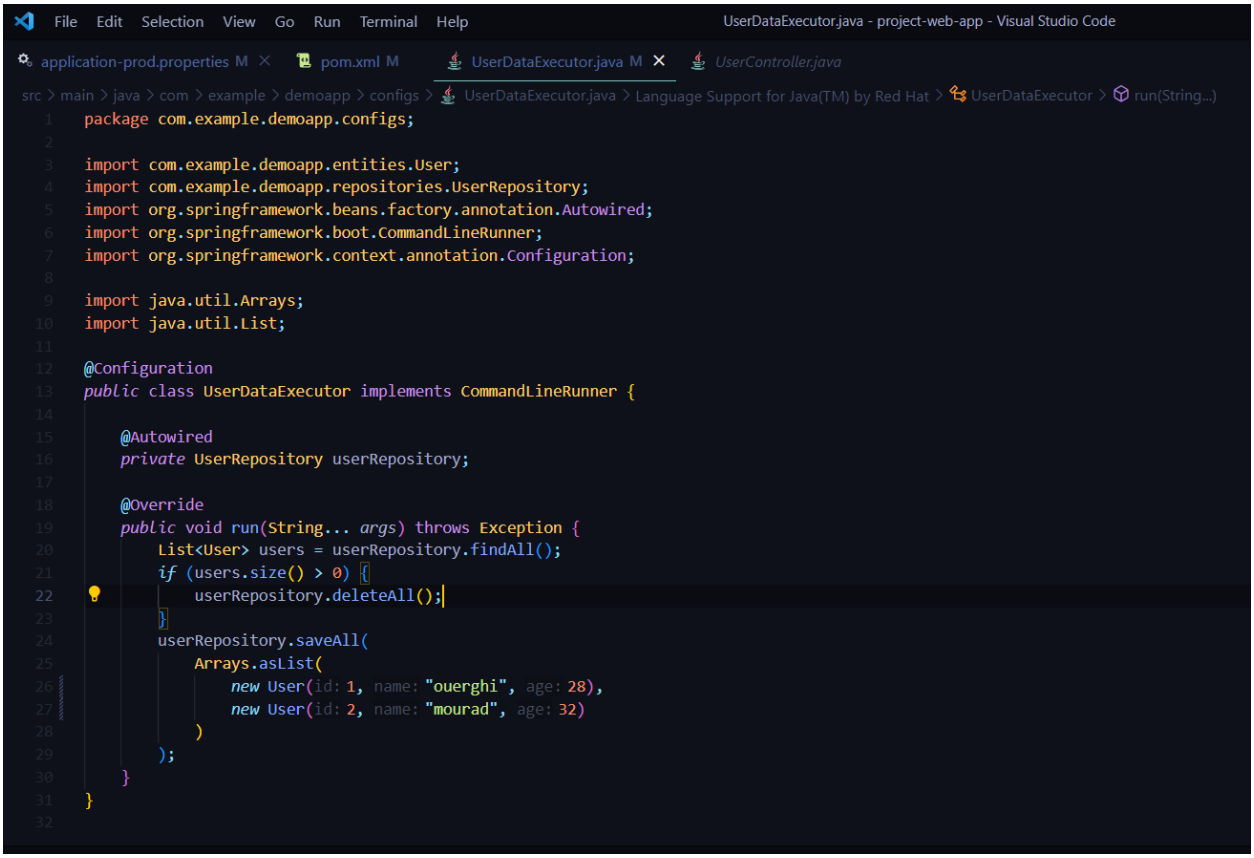
ii. Security Group

C’est le security group du "projet-web-server" après avoir terminé le projet, contenant une trois inbound rules, la première sur le port 8080 en TCP avec le security group du load Balancer pour permettre au load balancer d’envoyer des health checks, la deuxième sur le port 22 avec ssh avec le security group du machine bastion pour permettre l'accès au machine bastion et la troisième sur le port 3306 en TCP avec le security group de l’instance rds pour recevoir les données transmises



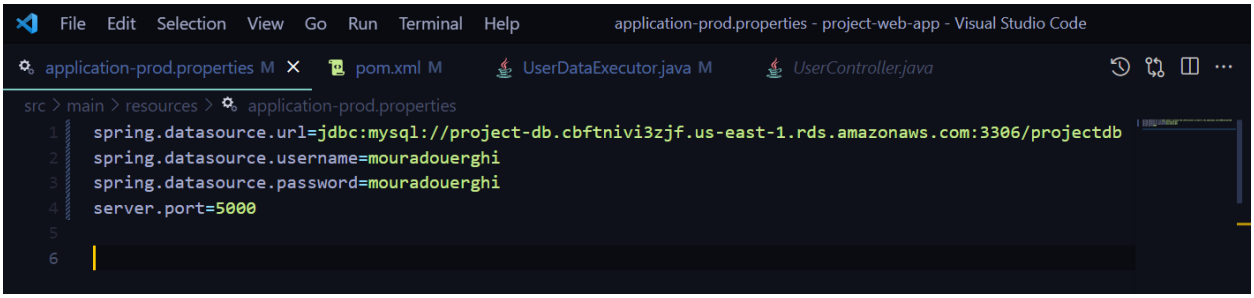
8. Déploiement de l’application web

C’est une simple application contenant une seul endpoint “/api/users” qui crée deux utilisateurs dans la base de données dès qu’elle se lance



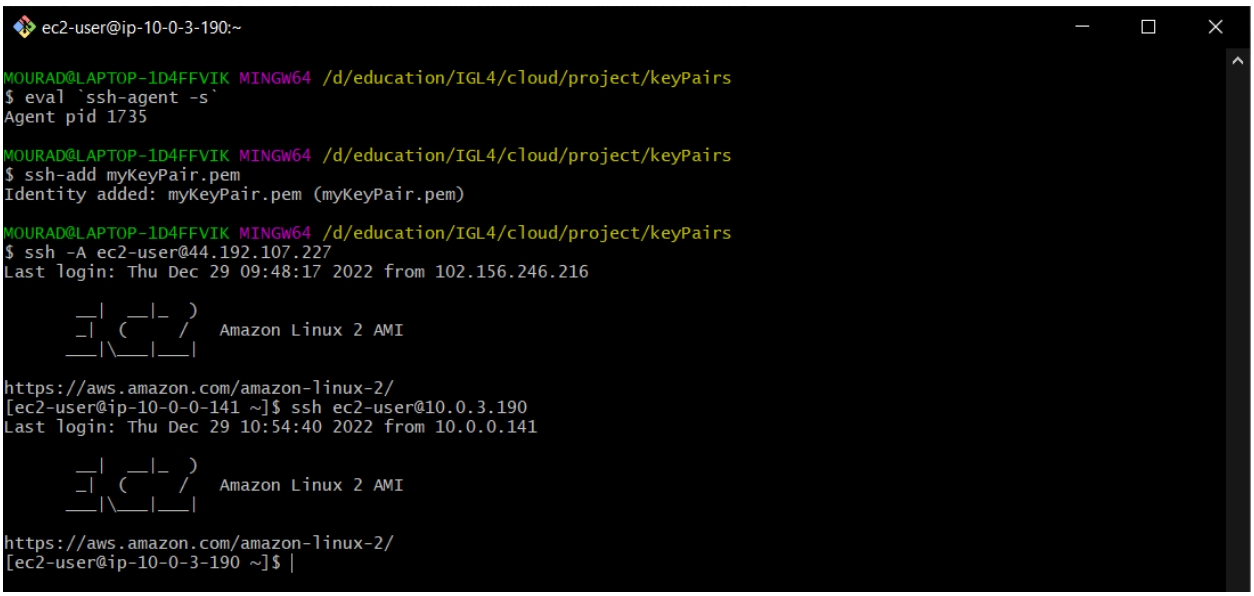
a. Liaison de l’application web avec l’instance RDS

Pour lier l’application avec la base de données rds créé j’ai mis l’url de l’instance rds le nom d'utilisateur et le mot de passe lorsque j’ai créé l’instance rds



b. Installation de l’environnement nécessaire

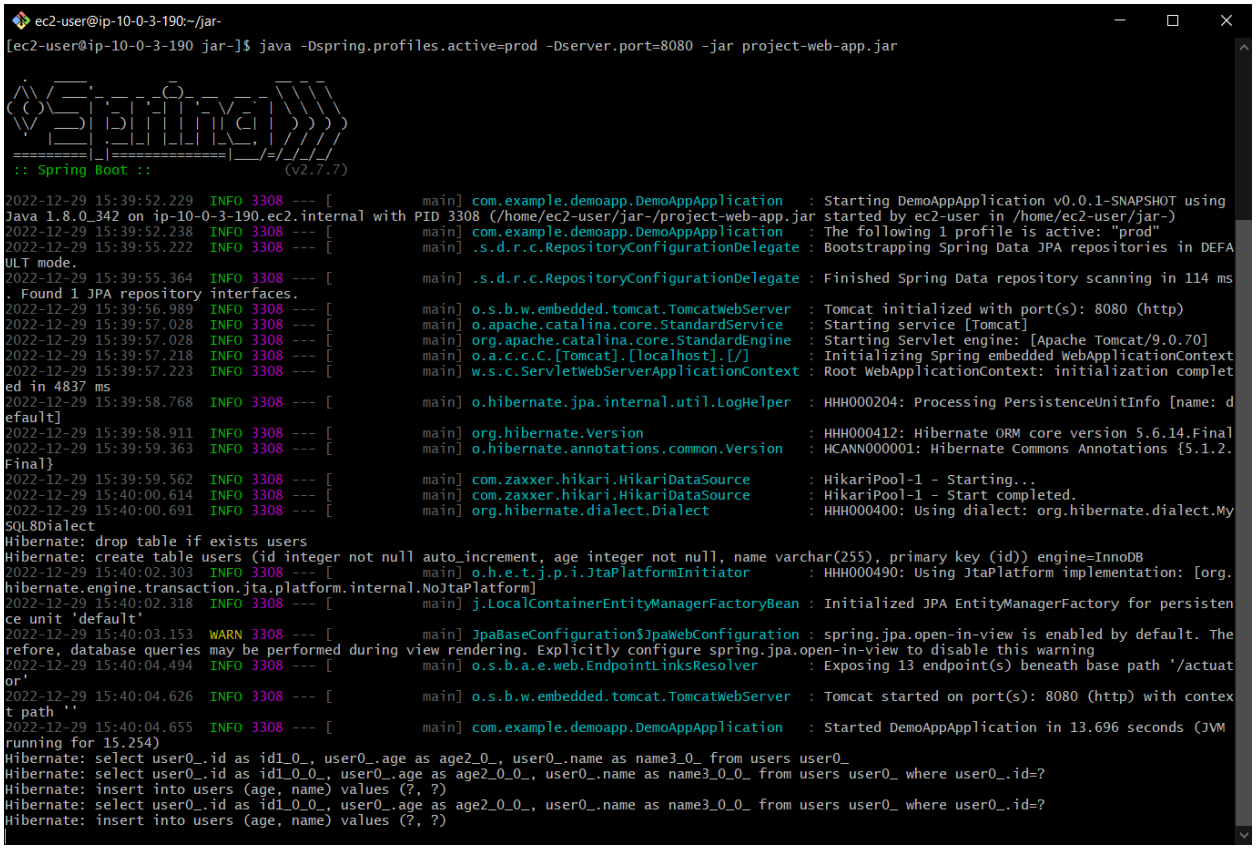
Pour se connecter à la machine bastion j’ai utilisé Git Bash, ensuite j’ai exécuter les commandes suivantes pour avoir l'accès ssh à la machine.



Ensuite j'ai installé git en utilisant la commande **"sudo yum install git"** puisque j'ai mis le fichier ".jar" dans une repository dans github pour faciliter la déploiement, ensuite j'ai installé jdk en utilisant la commande **"sudo yum install java-1.8.0-openjdk"** ensuite j'ai fait un **"git clone"** du fichier ".jar". J'ai oublié de faire des captures pour ces étapes donc j'ai essayé de faire une description détaillé

c. Déploiement de l'application web dans l'instance EC2

Comme le montre la figure ci dessous l'application se lance correctement et se connecte avec l'instance rds et crée les deux utilisateurs

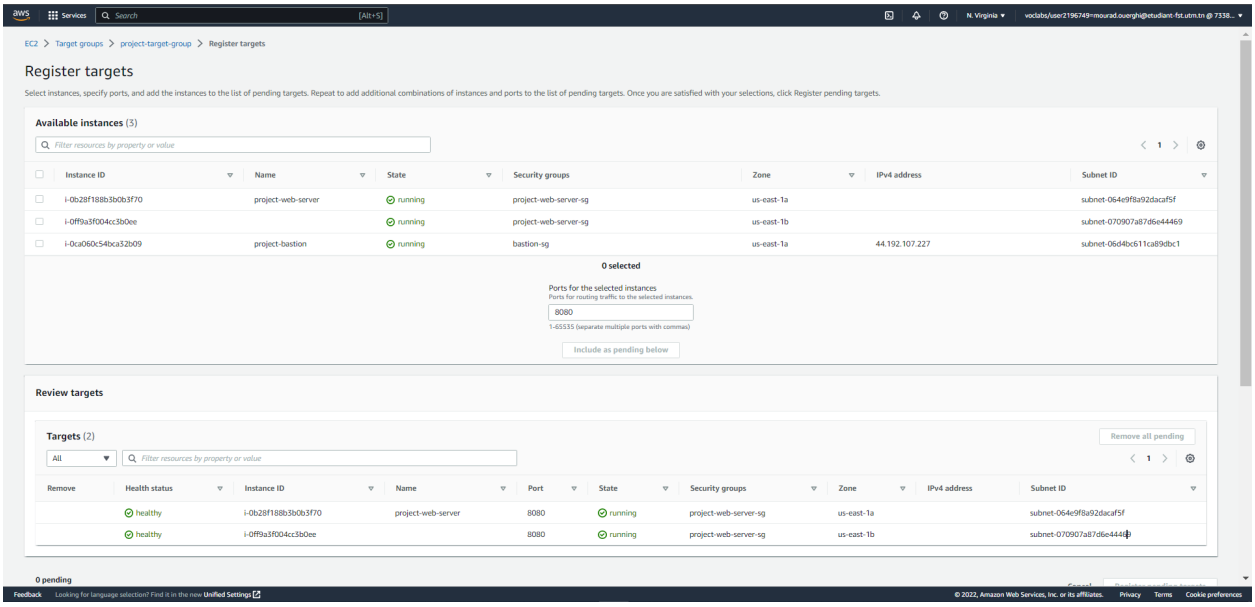


9. Création du load balancer

a. Création du Target Group

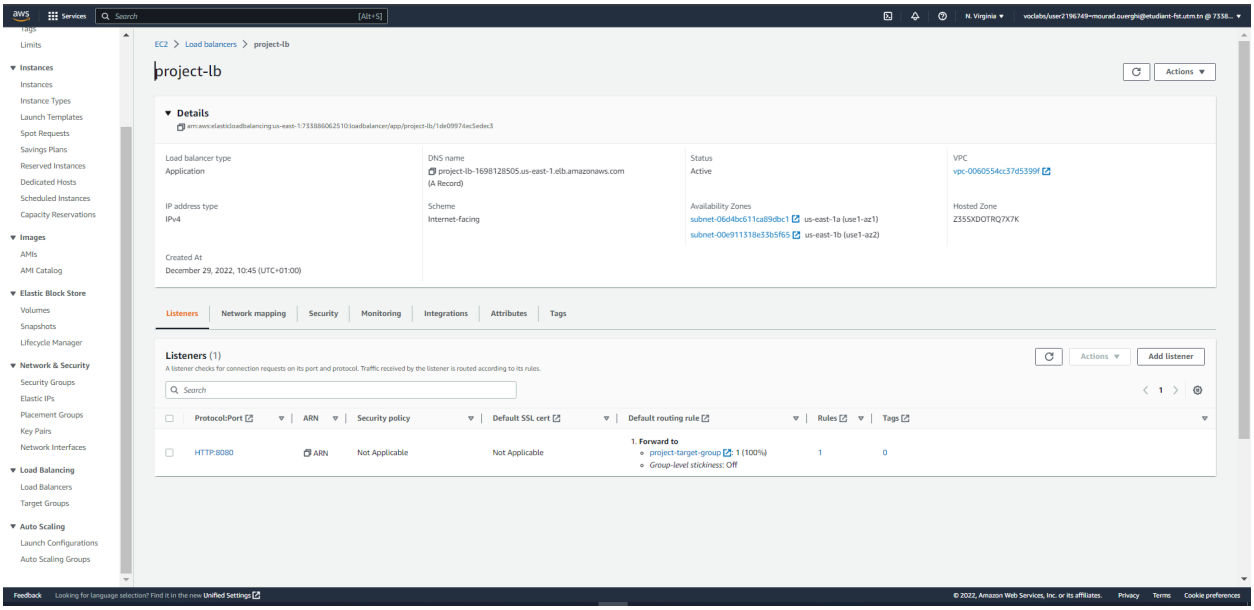
Création d’un target Group et enregistrement de l’instance “project-web-server”

(J’ai oublié de prendre une capture lorsque j’ai créé le target donc j’ai pris une capture après avoir terminer le projet)



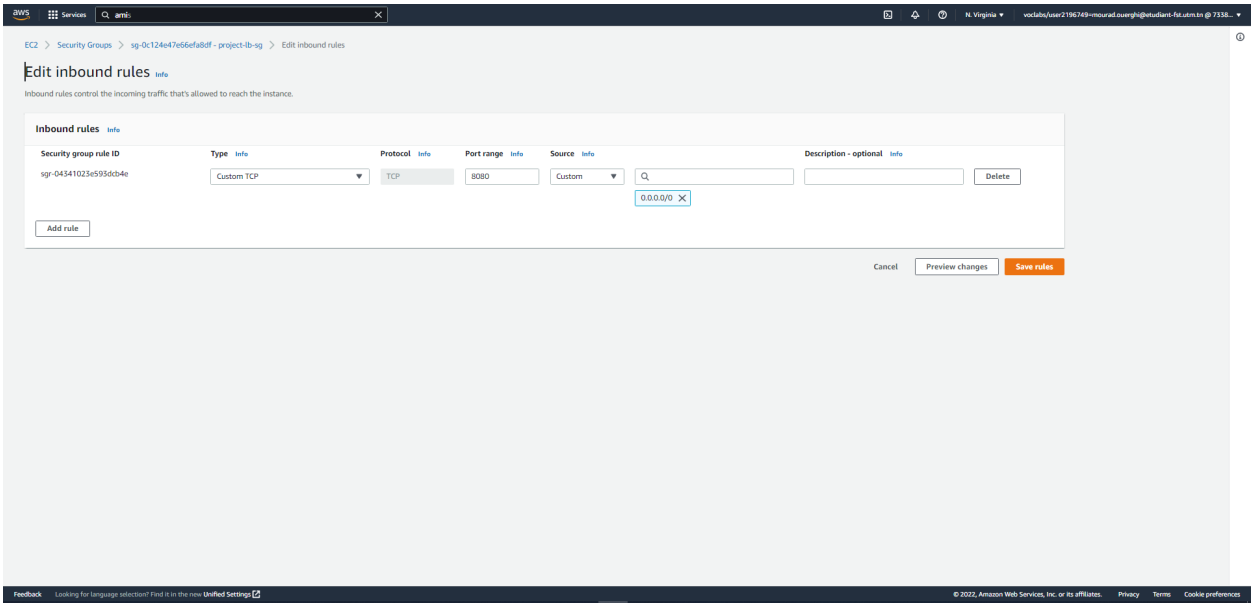
b. LoadBalancer

Création d’un load balancer sous le nom de “project-lb” sur les deux public subnets des deux availability zones

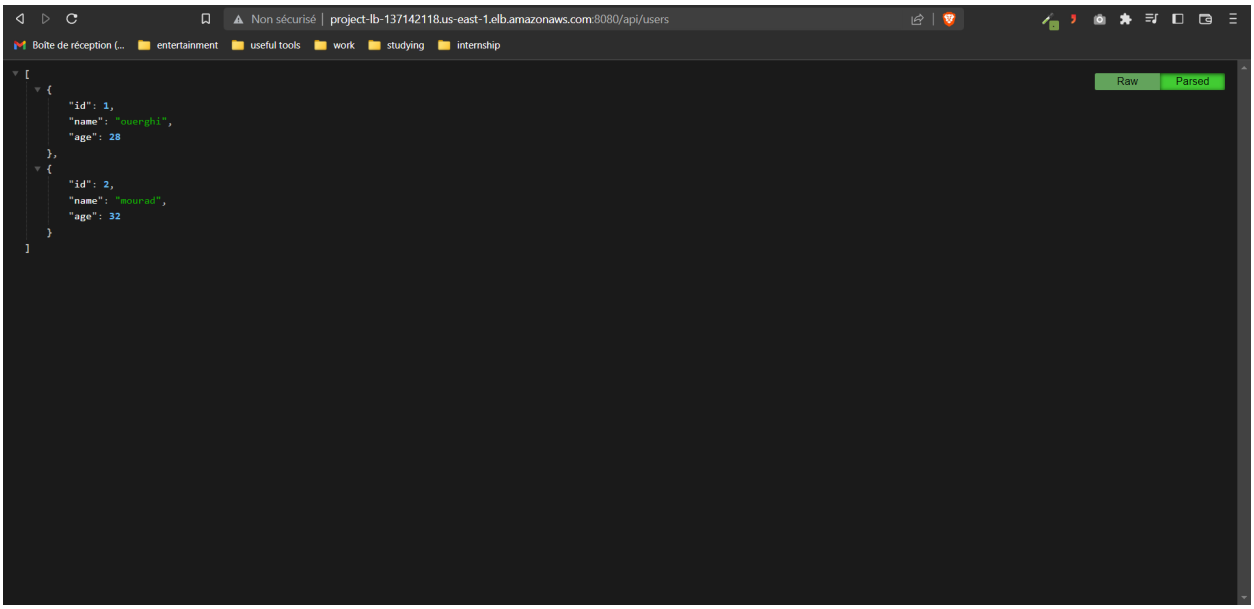


c. Security Group

Pour le security group j'ai ouvert seulement le port 8080 puisque l'application se lance sur le port 8080



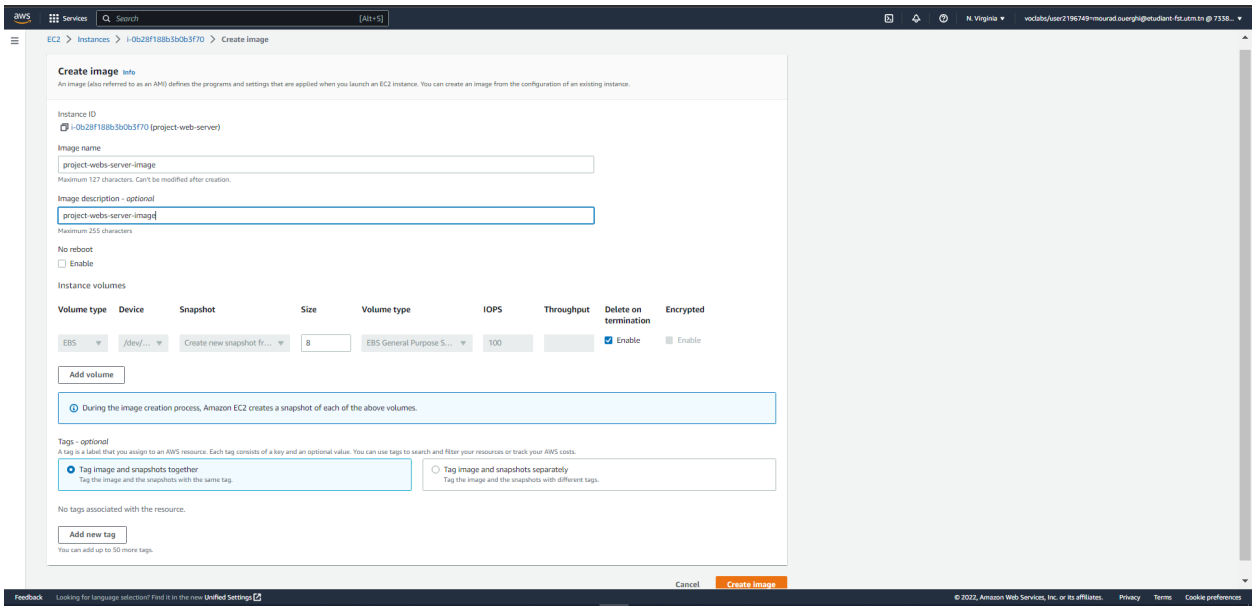
L'application est accessible comme le montre la figure ci dessous



10. Création du AutoScaling Group

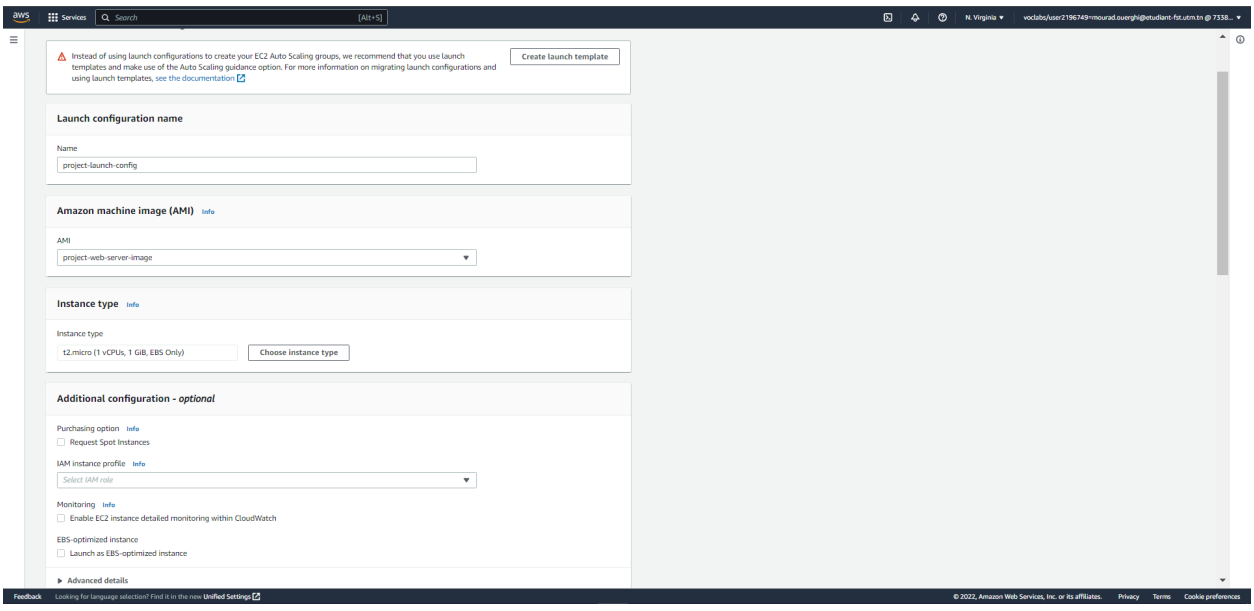
a. Création du Launch Configuration

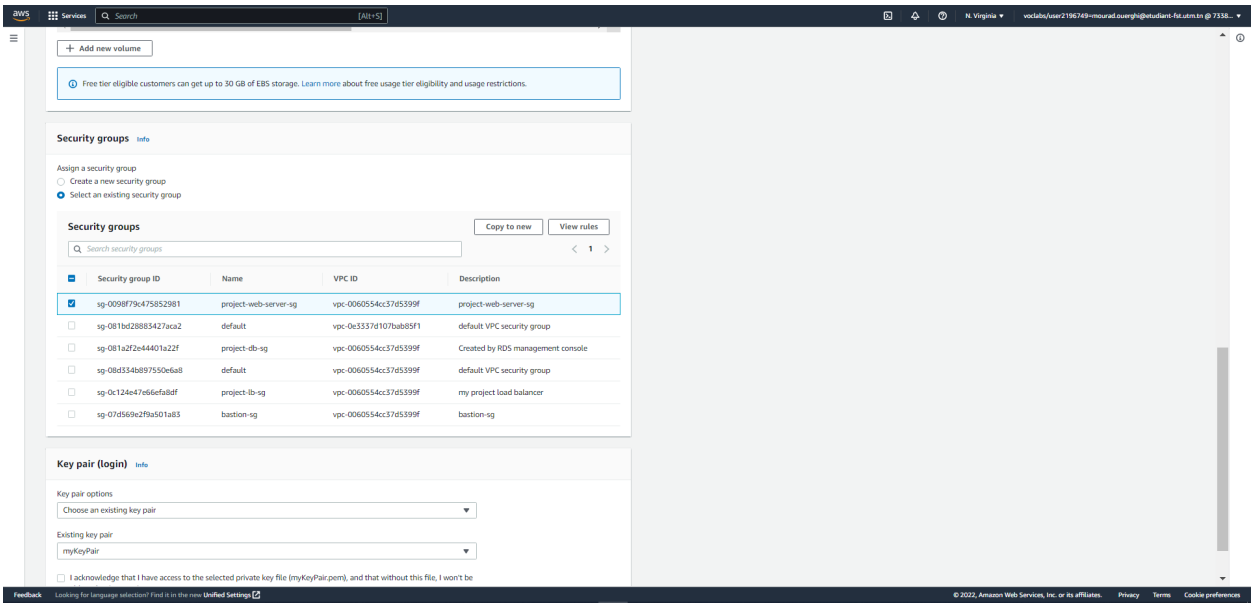
i. Création d'une AMI a partir du web server



ii. Création du Launch Configuration

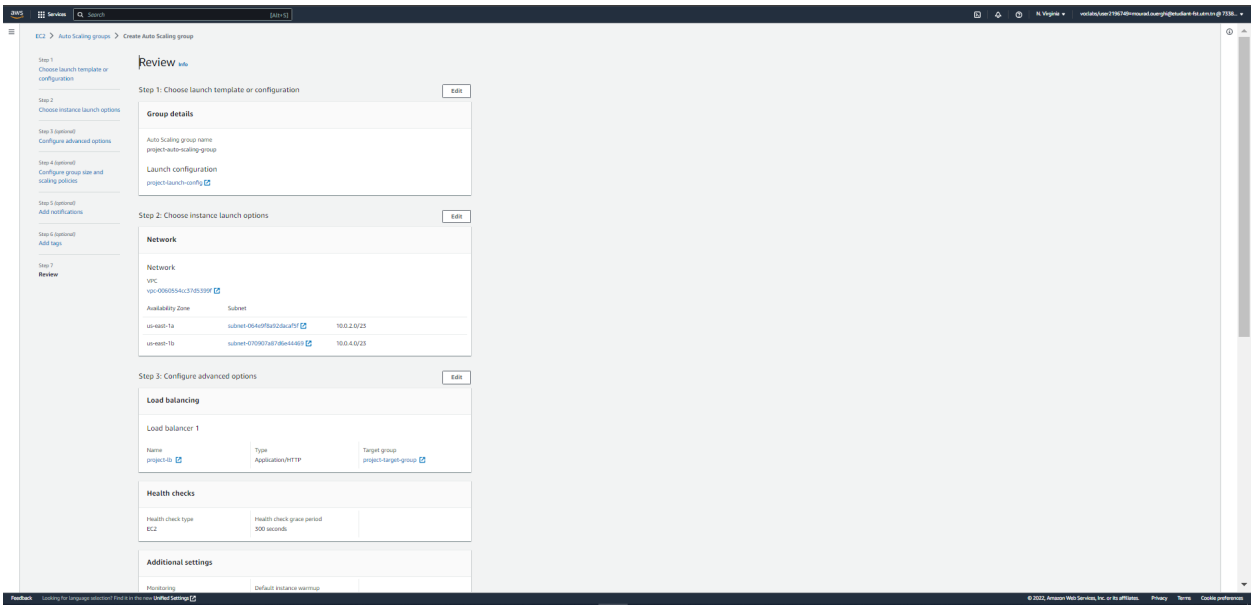
Création d'un launch config sous le nom de "project-launch-configuration" en utilisant l'image du "project-web-server" créé

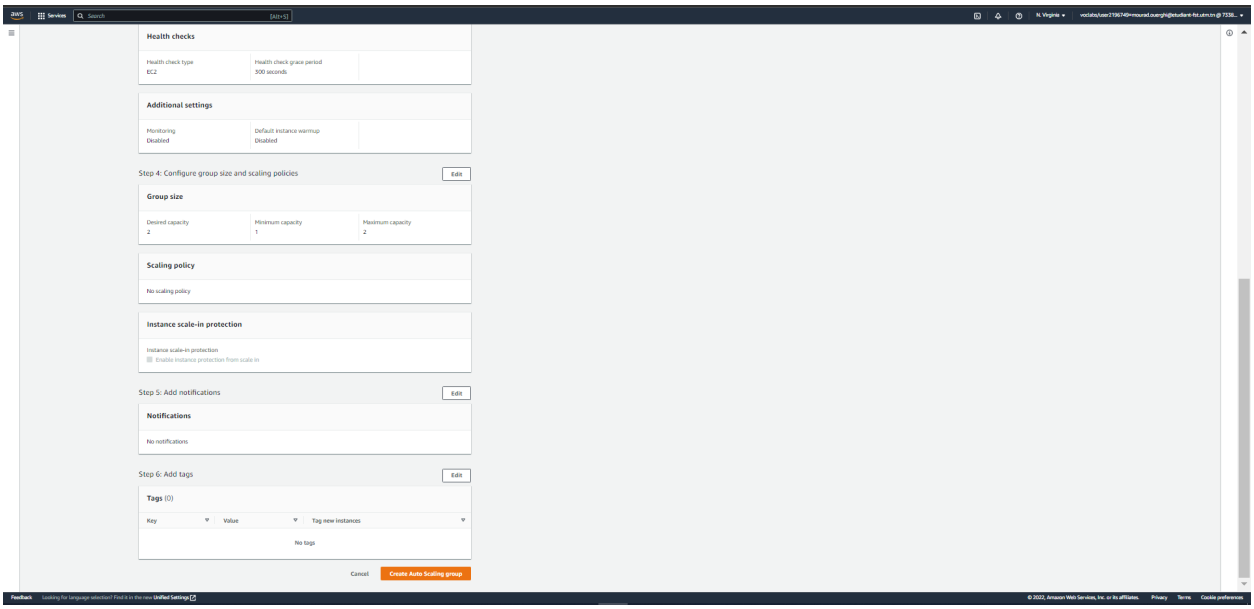




b. Création du AutoScaling Group

Création d'un auto scaling group sur les deux private subnets en utilisant le "project-launch-configuration" avec une desired capacity de 2 instances (je l'ai diminuer ensuite vers 1 instance puisque l'instance "project-web-server est déjà lancé")





La figure suivante montre que les deux target sont “healthy” donc le load balancer peut distribuer la charge entre les deux instances d'où la haute disponibilité de l’application