CAPSTONE PROJECT DOCUMENTATION

Cloud - AWS

Date : 08/04/2024

—

Zitouni Iptissem

I. Introduction

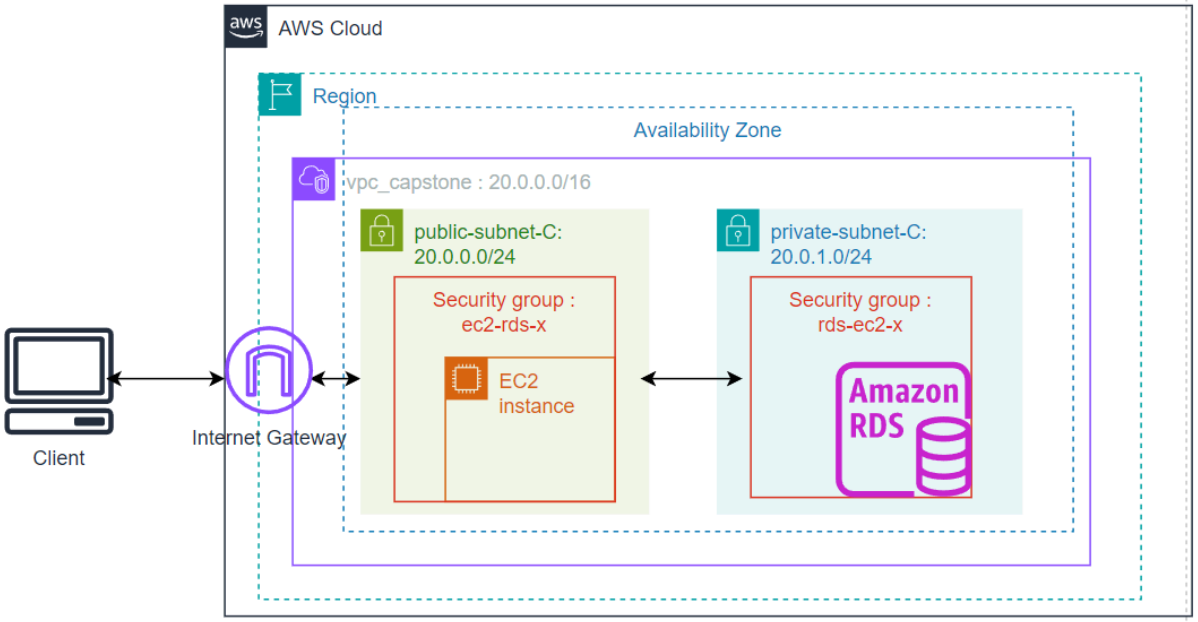
Through this project, we will deploy our website named capstone on the cloud and create a database.

To do this, we will go through several steps: an architecure diagram, a virtual private network and site hosting.
Then, creating a database and we will establish an SSH connection.

The technologies used to carry out this project: AWS cloud, like EC2, VPC, RDS..
To carry out this project, we use the sandbox functionality of our AWS sessions.

II. Diagram architecture

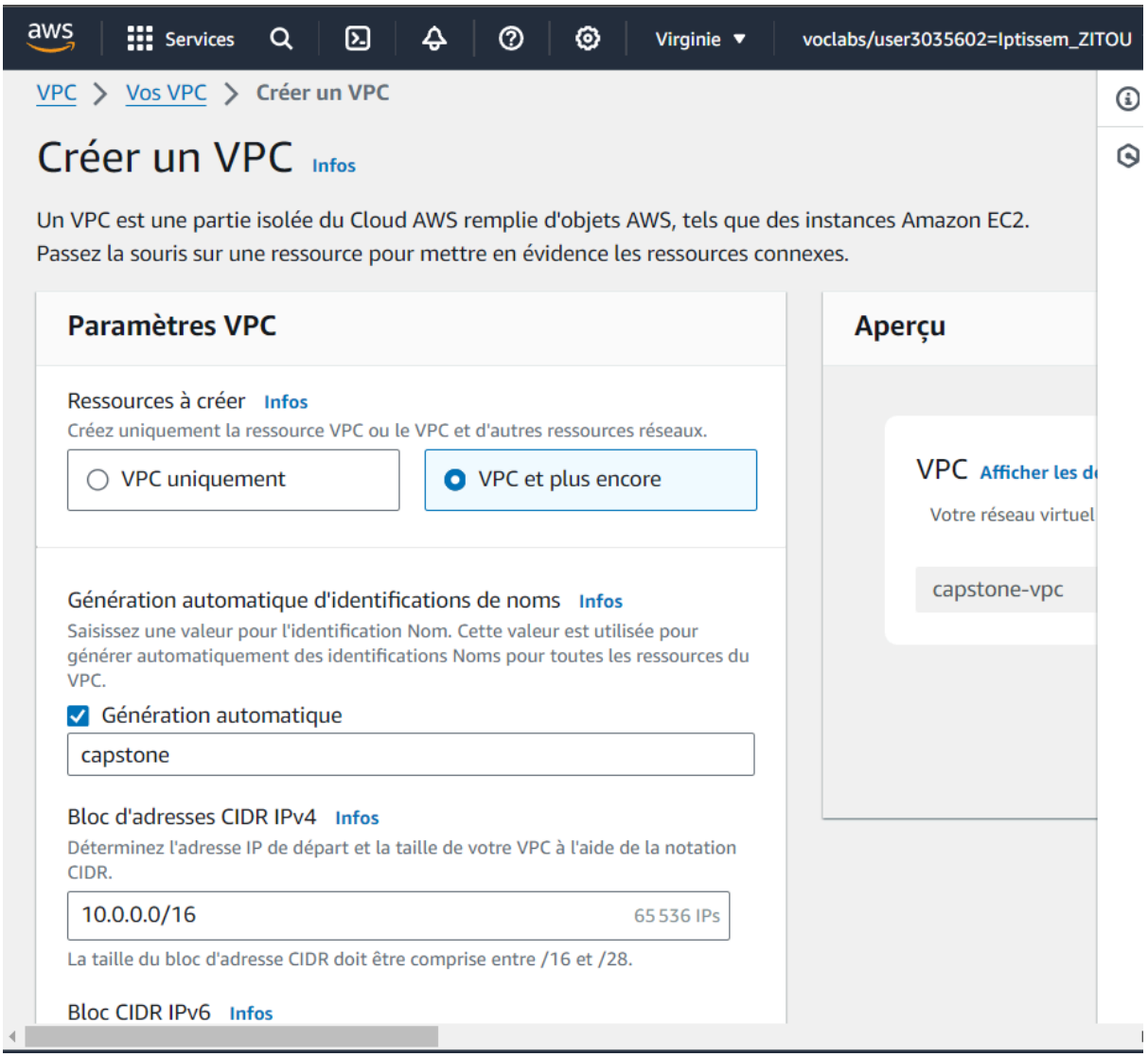
We have in our architecure a VPC with two subnets: a public in which will be our web server and the other private for the database.
We also have a client who will allow us to test our architecture.

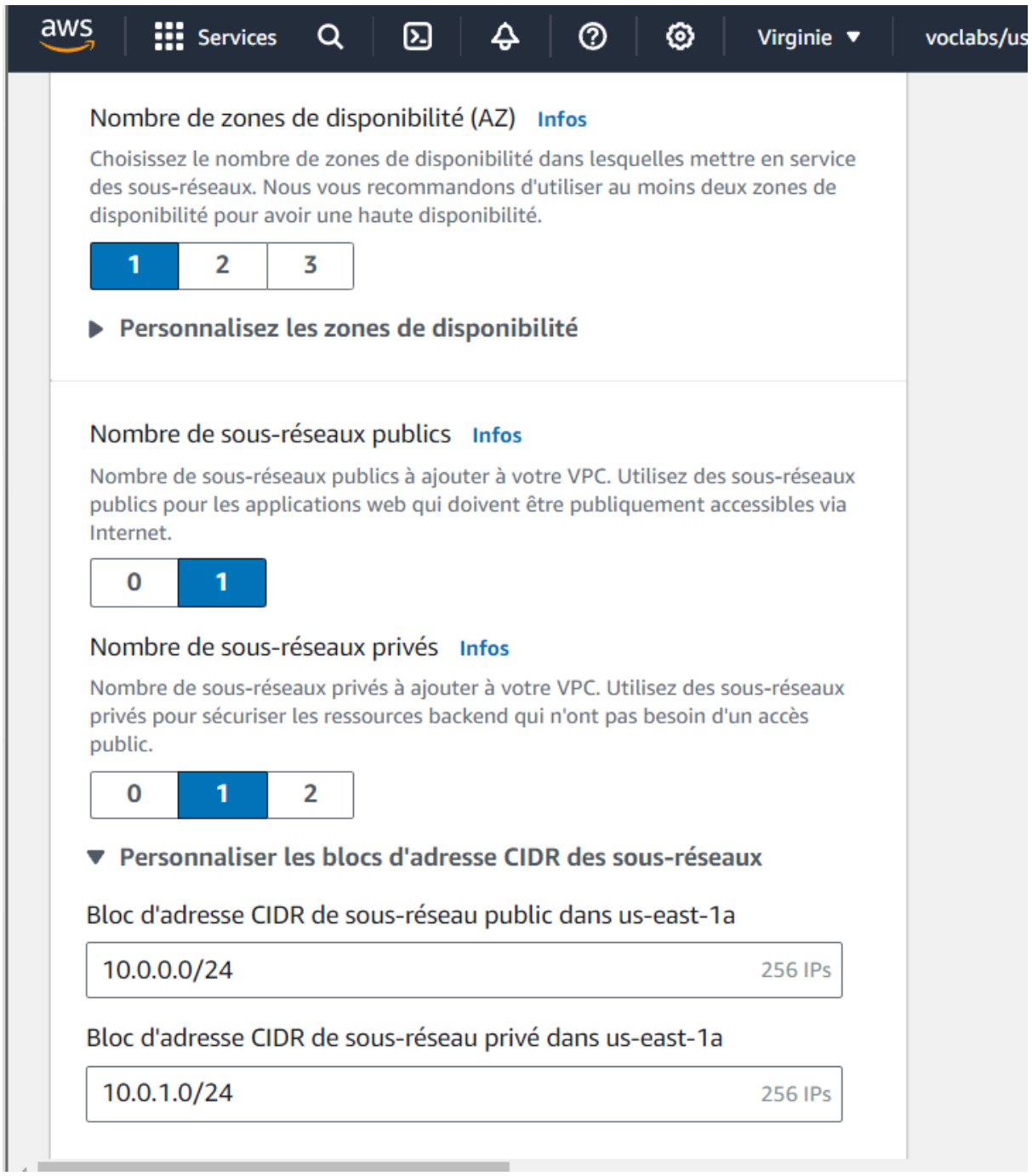


III. Deploying Architecture

1. Virtual Private Cloud VPC

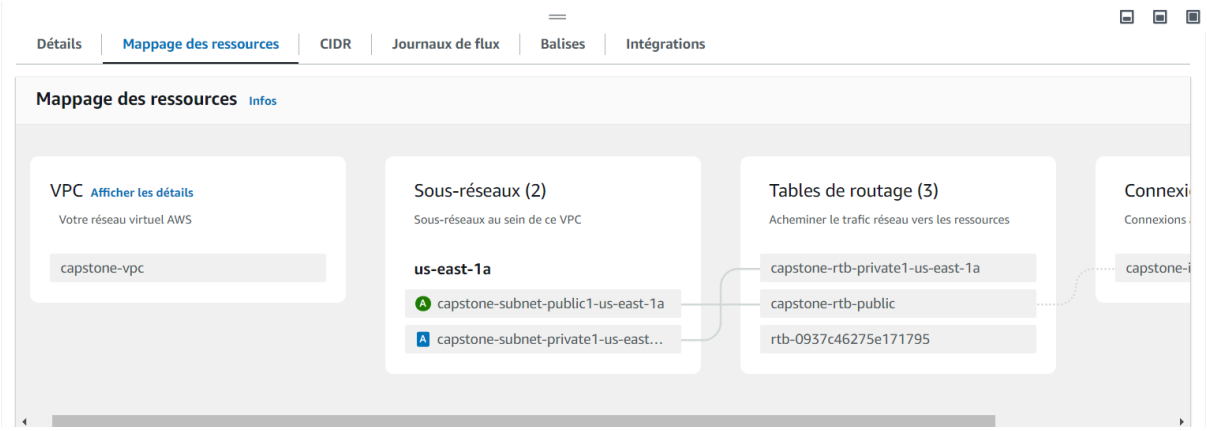
We configure our VPC to respect the scheme


aws
Services
Virginie
voclabs/user3035602=Iptissem_ZITOU
VPC > Vos VPC > Créer un VPC
Créer un VPC Infos
Un VPC est une partie isolée du Cloud AWS remplie d'objets AWS, tels que des instances Amazon EC2.
Passez la souris sur une ressource pour mettre en évidence les ressources connexes.
Paramètres VPC
Ressources à créer Infos
Créez uniquement la ressource VPC ou le VPC et d'autres ressources réseaux.
VPC uniquement
VPC et plus encore
Génération automatique d'identifications de noms Infos
Saisissez une valeur pour l'identification Nom. Cette valeur est utilisée pour générer automatiquement des identifications Noms pour toutes les ressources du VPC.
Génération automatique
capstone
Bloc d'adresses CIDR IPv4 Infos
Déterminez l'adresse IP de départ et la taille de votre VPC à l'aide de la notation CIDR.
10.0.0.0/16
65 536 IPs
La taille du bloc d'adresse CIDR doit être comprise entre /16 et /28.
Bloc CIDR IPv6 Infos
Aperçu
VPC Afficher les d
Votre réseau virtuel
capstone-vpc


aws
Services
Virginie
voclabs/us
Nombre de zones de disponibilité (AZ) Infos
Choisissez le nombre de zones de disponibilité dans lesquelles mettre en service des sous-réseaux. Nous vous recommandons d'utiliser au moins deux zones de disponibilité pour avoir une haute disponibilité.
1
2
3
Personnalisez les zones de disponibilité
Nombre de sous-réseaux publics Infos
Nombre de sous-réseaux publics à ajouter à votre VPC. Utilisez des sous-réseaux publics pour les applications web qui doivent être publiquement accessibles via Internet.
0
1
Nombre de sous-réseaux privés Infos
Nombre de sous-réseaux privés à ajouter à votre VPC. Utilisez des sous-réseaux privés pour sécuriser les ressources backend qui n'ont pas besoin d'un accès public.
0
1
2
Personnaliser les blocs d'adresse CIDR des sous-réseaux
Bloc d'adresse CIDR de sous-réseau public dans us-east-1a
10.0.0.0/24
256 IPs
Bloc d'adresse CIDR de sous-réseau privé dans us-east-1a
10.0.1.0/24
256 IPs

The description matches well: our vpc is created.

Here, the resource mapping section allows us to check that everything is good, it is.

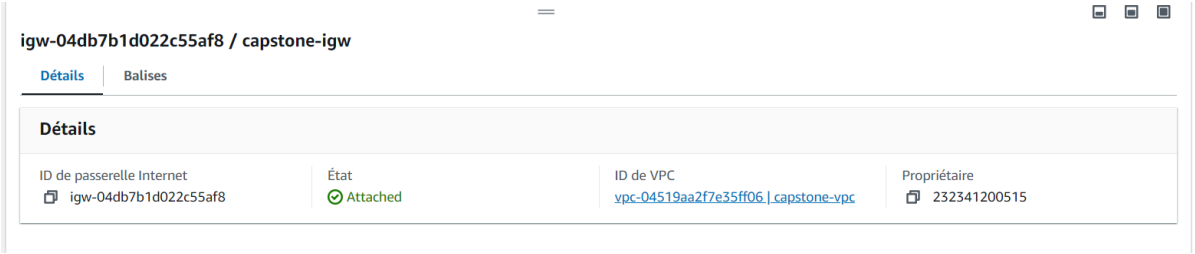


2. Routing and internet gateway

The VPC was created, it configured two very important steps: **the routing table** and the **internet gateway**.

Indeed, during the configuration of our VPC, we specified our subnetworks, so it deduces its parameters

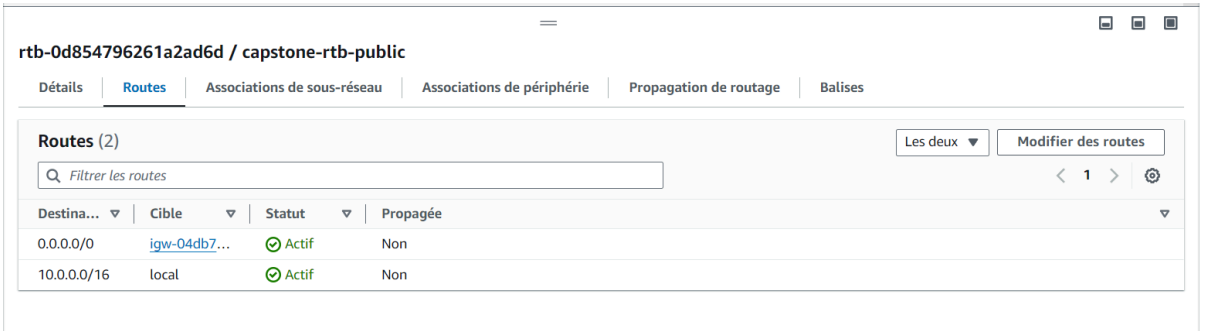
Internet gateway allows communication between resources within a virtual private cloud (VPC) and the Internet, enabling inbound and outbound traffic to and from the Internet.



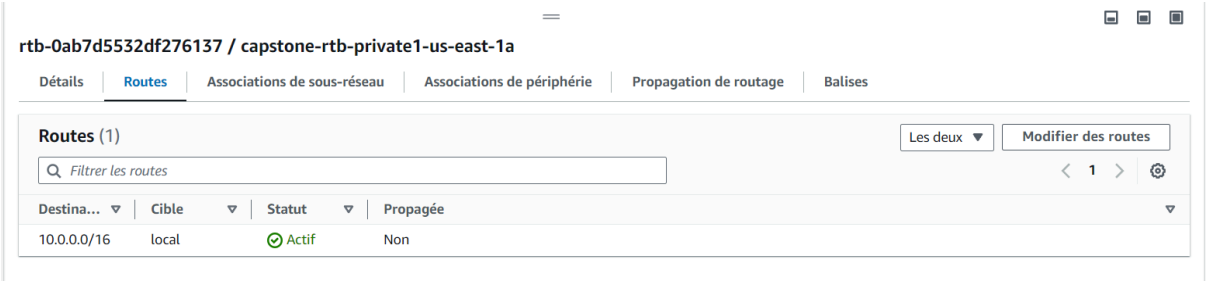
Routing table : set of rules that guides network traffic between different resources in an AWS virtual network, indicating where to send traffic based on its destination.

Public routing table : routes traffic to resources accessible from the Internet

Here, the 0.0.0.0/0 means internet.



Private routing table : directs traffic to resources internal to the virtual network



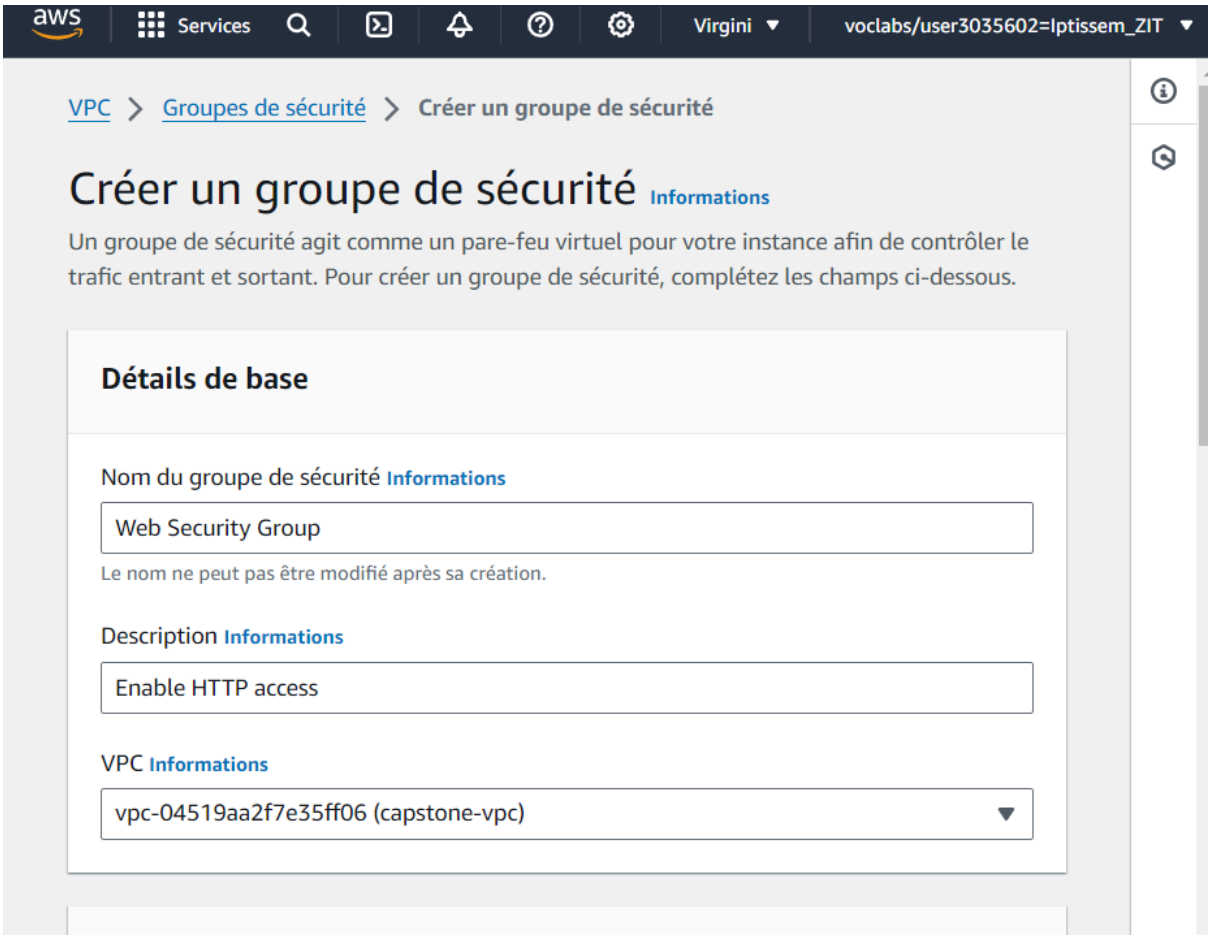
The 10.0.0.0/16 is the local route.

3. Creating security group

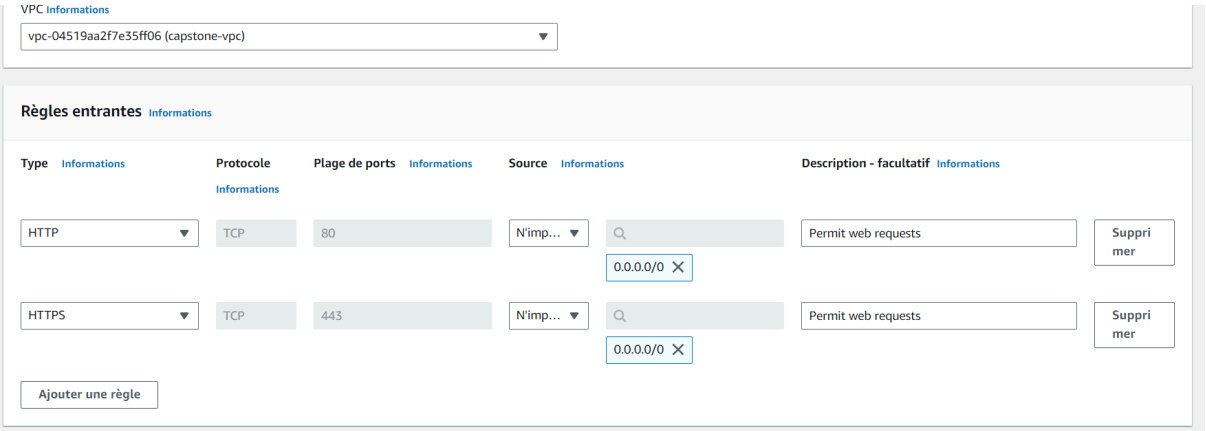
After configuring our environment, we will create security groups for our 2 subnets. création groupe de sécurité.

They acts as a virtual **firewall** for instances, controlling inbound and outbound traffic based on defined rules, thus helping to enhance network security within the AWS environment.

- Group security for web equip :



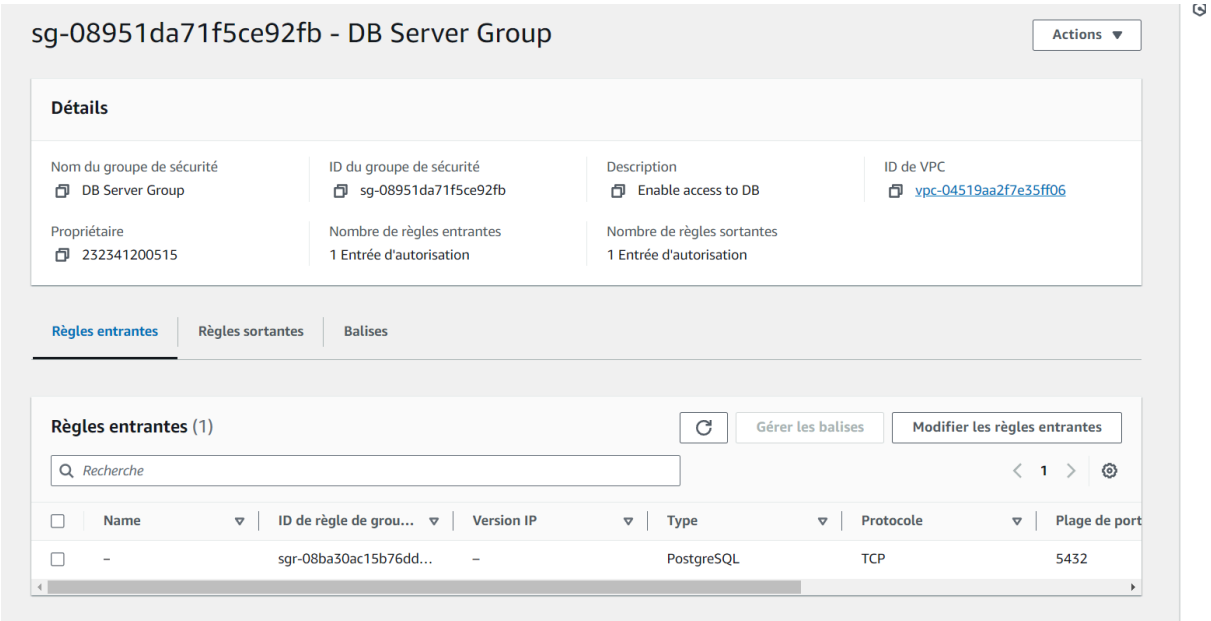
- Rules we added : HTTPS, HTTP



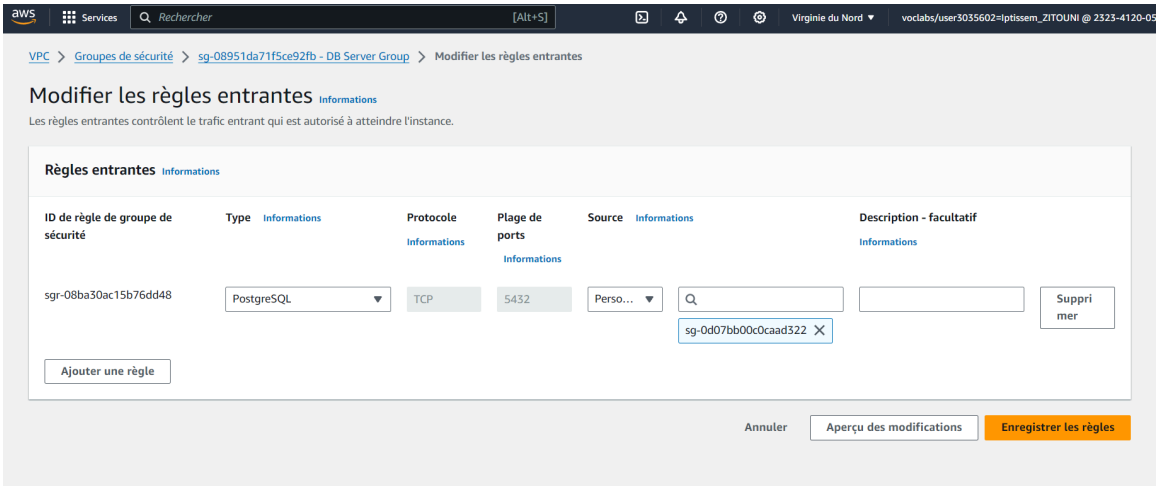
We will add the incoming **SSH** rule to be able to access our virtual machine

Well !

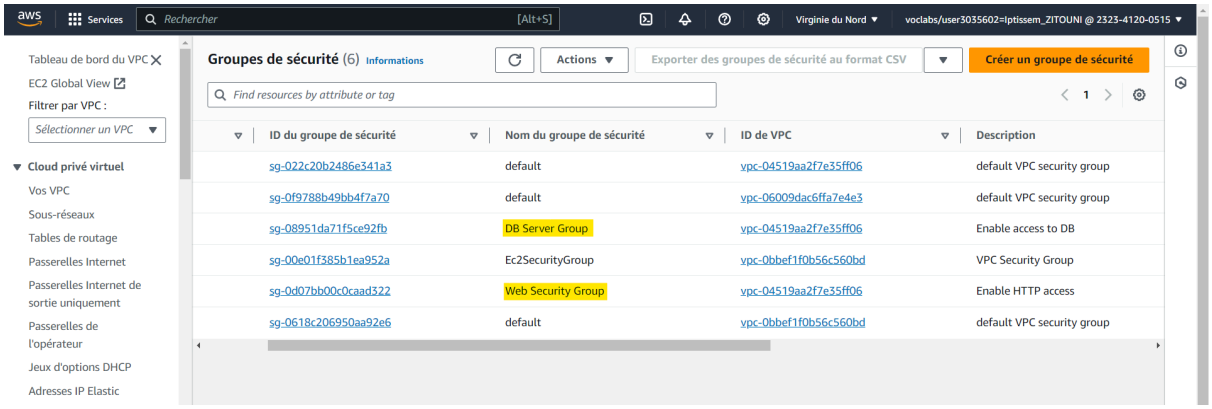
- Db server group : Permit access from Web Security Group



- Rules : We specify the web security group



Our security groups are well created and active :

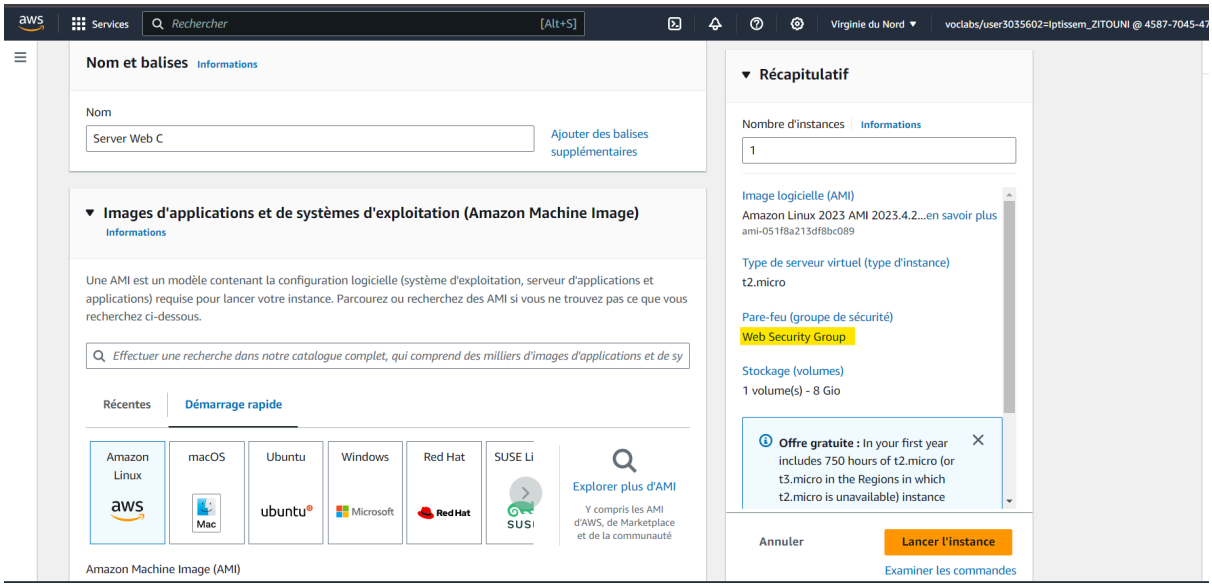


IV. Launch a Web Server Instance

Our instance is virtual machine configured to host websites or web applications, serving content to users accessing it through their web browsers over the Internet.

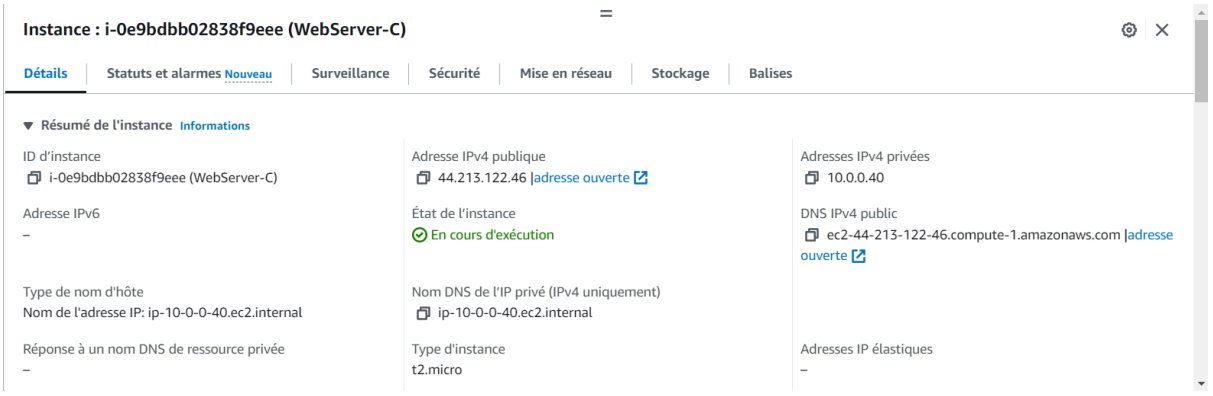
Here, we are going to deploy a website.

In our AWS console, we go to Instance/Launch Instance: there is more information to complete.
We assign to our server the web security group create earlier



Our server has been created.

A public IP address is assigned to enable users to access it over the Internet, facilitating the routing of requests from internet users to the server for web content delivery.
It is with this one that we will access our site



To deploy our site, we had two solutions: either with a scrpit, or with hands. We chose the second solution here.
We first install git to retrieve our files and httpd which is the apache server.

```
[ec2-user@ip-20-0-0-62 ~]$ sudo yum  install git
Last metadata expiration check: 0:12:37 ago on Mon Apr  8 14:10:52 2024.
Dependencies resolved.
================================================================================
```

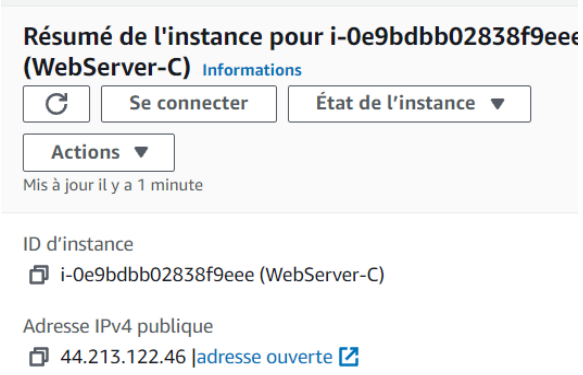
```
[ec2-user@ip-20-0-0-62 ~]$ sudo yum install httpd
Last metadata expiration check: 0:12:49 ago on Mon Apr  8 14:10:52 2024.
Dependencies resolved.
================================================================================
 Package              Arch     Version                    Repository     Siz
================================================================================
```

Then, we git clone our project and copy our site in /var/www/html/

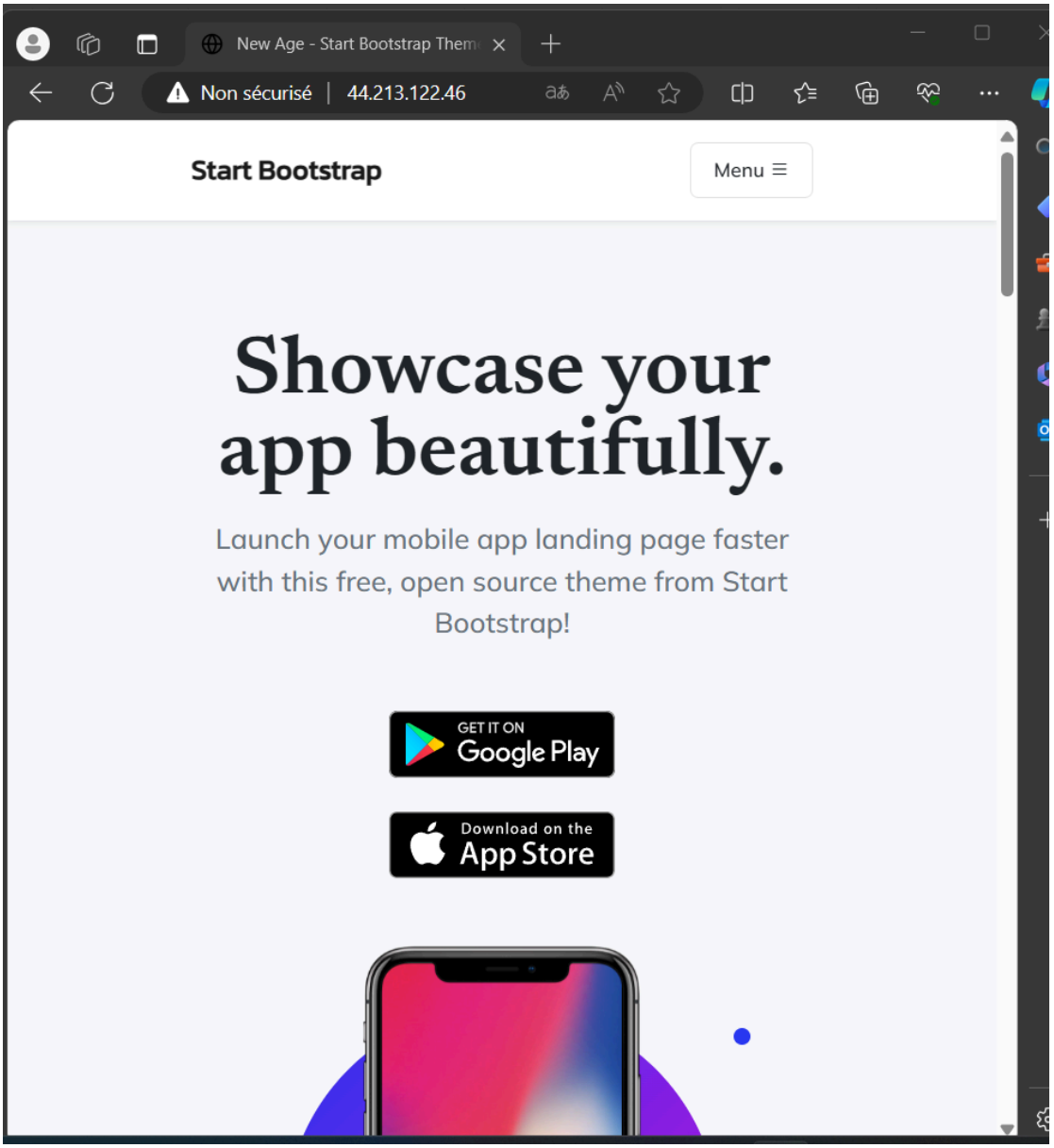
```
[ec2-user@ip-20-0-0-62 ~]$ git clone https://github.com/iptissem/capstone.git
Cloning into 'capstone'
```

```
[ec2-user@ip-10-0-0-40 capstone]$ ls
sample-app
[ec2-user@ip-10-0-0-40 capstone]$  cd sample-app/
[ec2-user@ip-10-0-0-40 sample-app]$ ls
assets  css  index.html  js
[ec2-user@ip-10-0-0-40 sample-app]$ sudo cp -r * /var/www/html/
[ec2-user@ip-10-0-0-40 sample-app]$ sudo systemctl start httpd
[ec2-user@ip-10-0-0-40 sample-app]$
```

We restart our httpd service and our web server is configured.
You can also check the server status by doing a systelctl status httpd: it is running.
We refresh our instance.



We put our IP in browser : succes !



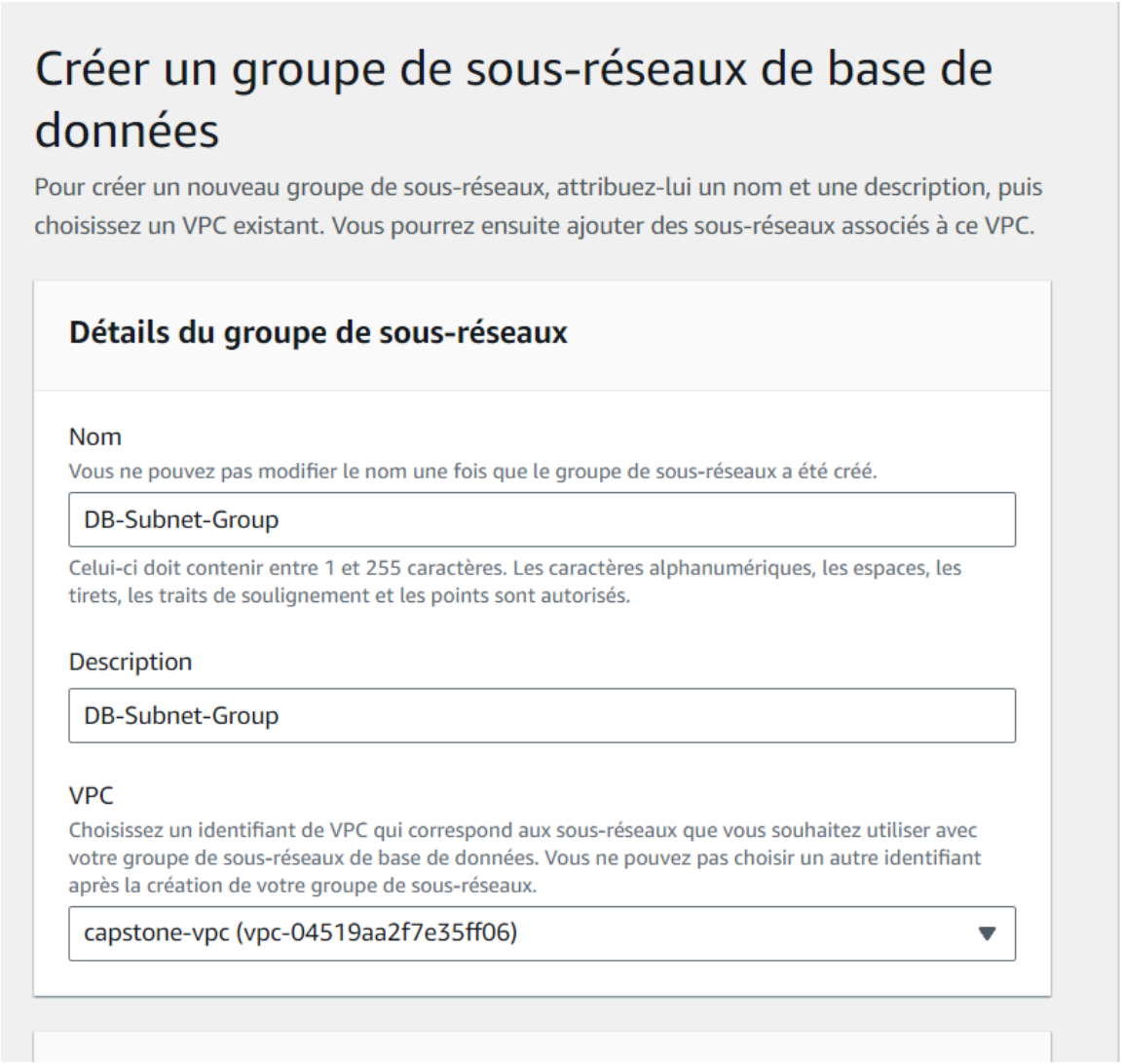
!

IV. Build our Database

Amazon RDS (Relational Database Service) is a managed database service that simplifies database setup, operation, and scaling in the cloud, offering support for various relational database engines.

We will use postgresSQL.

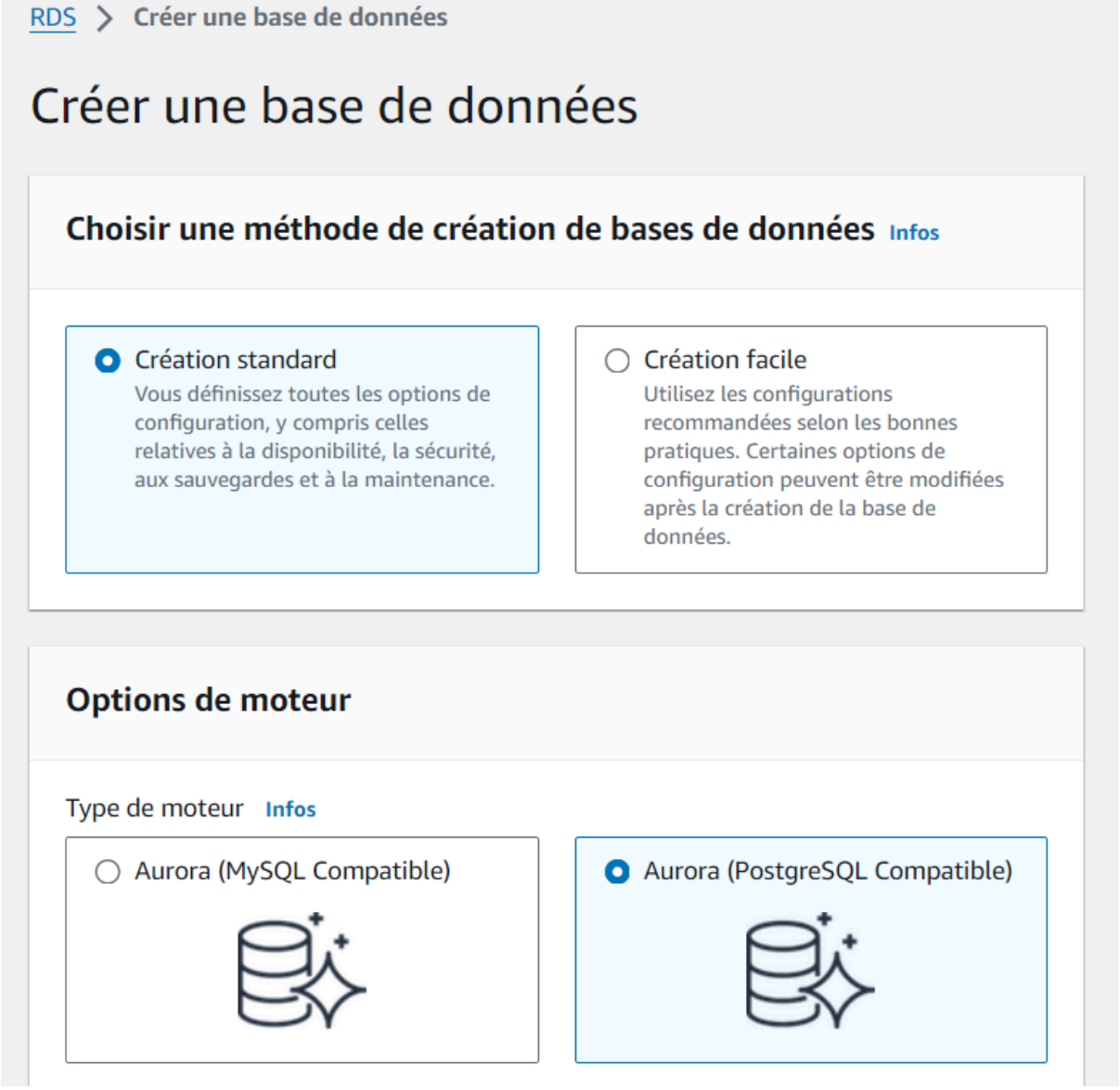
4. Create a DB Subnet Group



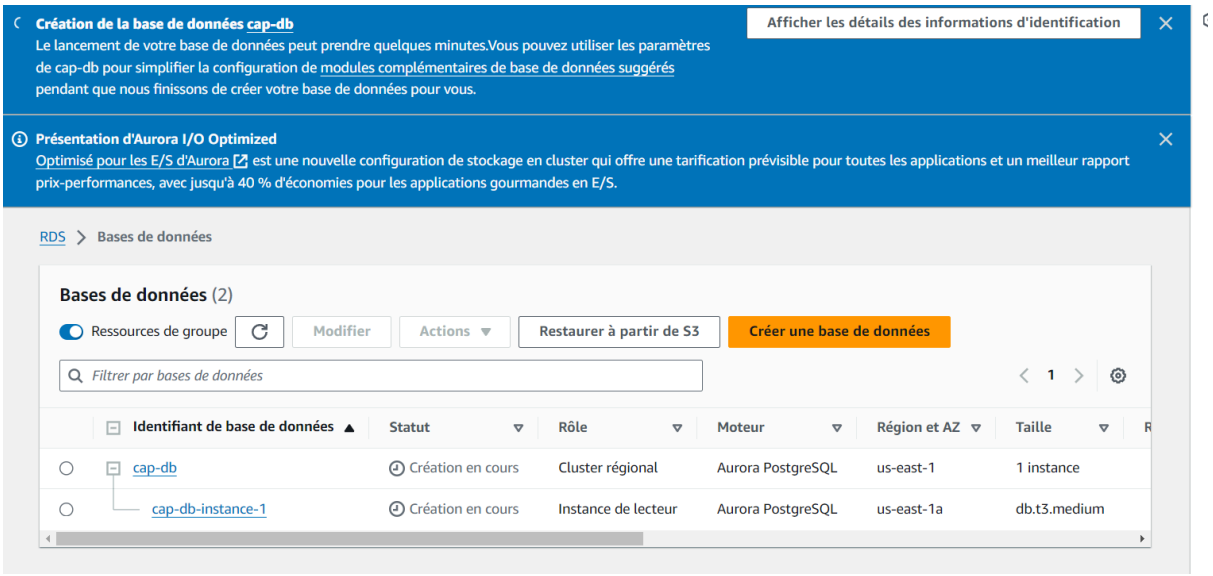
We select the subnets associated with the CIDR ranges **10.0.1.0/24**

5. Create an Amazon RDS DB Instance

database creation :



We then configured our fields, the most important :
DB instance identifier: cap-db
Initial database name: cap
password
VPC
Make sure your database is public.

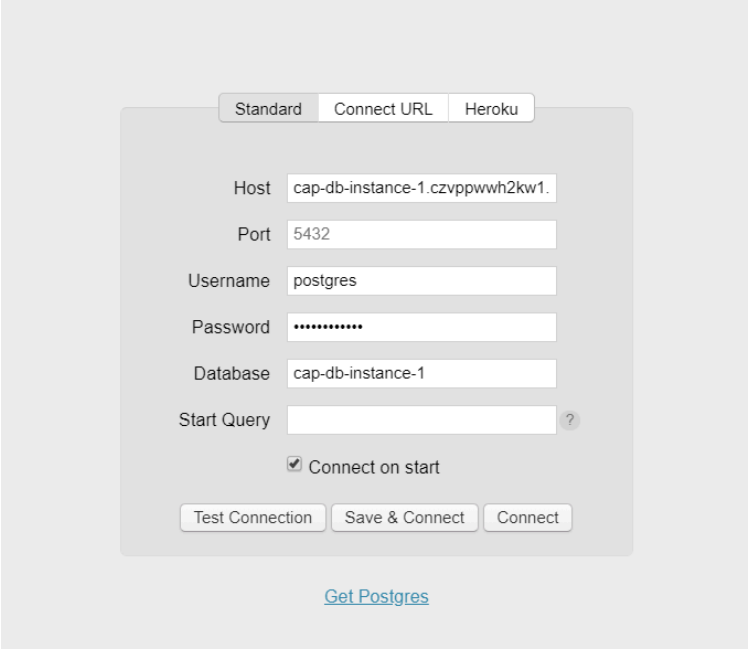


Our database has been created.

V. Testing Database

For testing our job we are going to use postbird : a PostgreSQL client application that provides a user-friendly interface for managing PostgreSQL databases and executing queries.

For these we have to find our endpoint in cap-db-instance-1. :
cap-db.cluster-czvppwwh2kw1.us-east-1.rds.amazonaws.com



VI. Conclusion

After several tests, we can not reach our two hosts
Here are the tests done to try debbug:
Check our VPCs,
that the database is public,
We have carefully selected our networks and security group.