AWS CLOUD SOLUTION FOR 2 COMPANY WEBSITES USING A REVERSE PROXY TECHNOLOGY

It is important to note that this infrastructure set up would not be covered in the AWS free tier as we would be using resources that would come at a cost .it is therefore advisable to set up a budget cost and configure notification to a pre-defined limit. Once done with the infrastructures we should ensure to delete all resources once the project is completed .

Please also note these overall principles that are used by the project are AWS concepts but they are common across most of the major Cloud Providers (e.g., Microsoft Azure and Google Cloud Platform).

This project would be implemented manually as it is important to understand the manual concepts before we begin to automate and this is the best way for to start because automation might become very frustrating if you don't understand these AWS specific concepts manually.

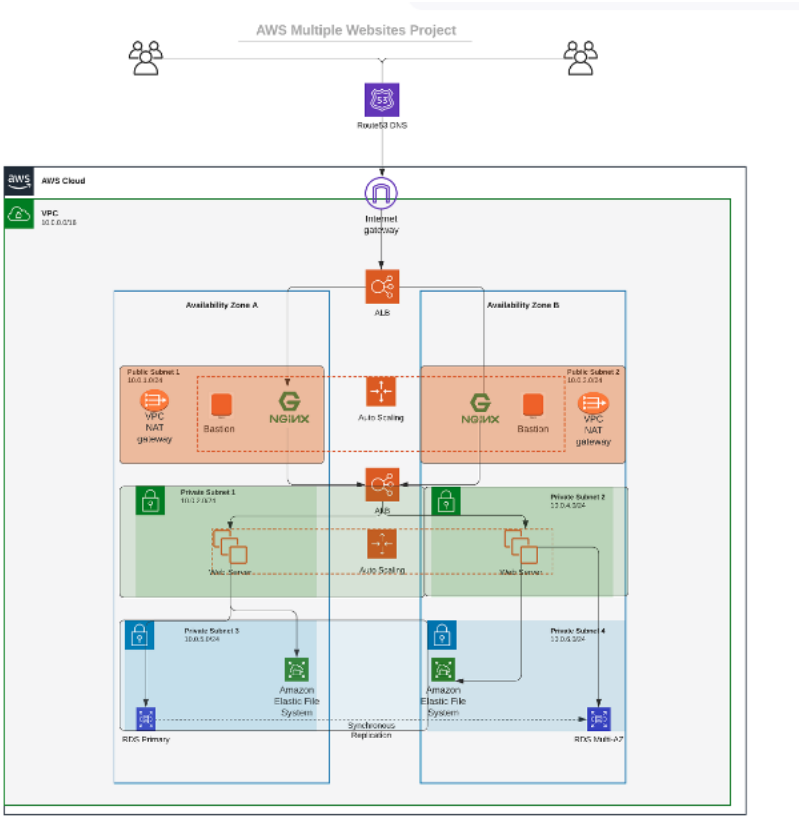
We will build a secure infrastructure inside AWS VPC (Virtual Private Cloud) network for a fictitious company named "PERFECT" that uses **WordPress CMS** for its main business website, and a **Tooling Website**

https://github.com/eyewande2022/perfect-project-config.git

This would be used for their DevOps team. As part of the PERFECT company's desire for improved security and performance, a decision has been made to use a reverse proxy technology from **NGINX** to achieve this.

Pre-requisites for this project are:
Extensive Knowledge of AWS infrastructure resources namely is required for this project:
Route53 ,Application Load Balancer Nginx Reverse Proxy
VPC, Private and Public Subnets, Bastion Server ,DNS
Elastic File System, RDS, Key Management Systems and many more:

Cost, Security, and Scalability are the major requirements for this project. Hence, implementing the architecture designed below, ensure that infrastructure for both websites, WordPress and Tooling, is resilient to Web Server's failures, can accommodate to increased traffic and, at the same time, has reasonable cost.



AWS MULTIPLE WEBSITE PROJECT

EXPLAINING EACH COMPONENT IN THIS INFRASTRUCTURE

1) Route 53 DNS server : provided by AWS helps us to resolve our ip address to a hostname. e.g mapping out our IP address to pftd.com we are using in this project .The DNS server helps us achieves that

2) Internet Gateway is what connected the internet or external network to our VPC (Virtual Private Cloud).Entry point to our VPC.

3) ALB: A single point of contact that helps us distribute traffic to multiple target server and this is where high

availability concept is used .we are using an internet facing ALB and an internal facing ALB for this project

4) VPC: Virtual Private Cloud Is an isolated environment in this cloud that has its own network pool addresses and can share its IP address with its component e.g 10.0.0.0/16

5) NGINX Reverse PROXY Server: It's a public server that sits behind a firewall in a private network and directs clients request to the appropriate backend server (2nd ALB internal). It gives an additional level ion abstraction and control to ensure smooth flow of network traffics between clients and servers

6) Autoscaling group: Helps us in scaling up and down our instances:

Please note in this project we would be using the autoscaling group to create the following

a) At public subnet level our Bastion and NGINX reverse proxy has been placed in an ASG.
b) At the Private subnet level, we use it to create the servers for the tooling and WordPress

7) NAT Gateway: Helps our private subnet to communicate with the internet. Without this our internet cannot communicate with our private subnet .

8) Public Subnet allows incoming traffic into it through the internet gateway

9) Private subnet: does not allow incoming traffic from the public but can only accept through the NAT GATEWAY.

10) Amazon Elastic File System (Amazon EFS) provides a simple, scalable, fully managed elastic NFS file system for use with AWS Cloud services and on-premises resources. It is built to scale on demand to petabytes without disrupting applications, growing and shrinking automatically as you add and remove files, eliminating the need to provision and manage capacity to accommodate growth. It has a dynamic elasticity and instantly scales your file system storage capacity up or down as you add or remove files without disrupting your applications, giving you the storage, you need when you need it. It is fully managed by AWS

11) RDS (Relational Database Service) : This is a managed database system by AWS. It is easy set up ,operate and scale up RDS databases.

12) Bastion Server : It Is the entry point of all of our resources.

CONNECTIVITY OF ALL INFRASTRUCTURES EXPLAINED:

Steps:

a) Our clients on the internet hits that endpoint Wordpress.pftd.com.
b) It gets to the route53 which then resolves that name to the “DNS name of our Application Load Balancer.

This internet traffic comes in via the internet gateway to our ALB which is facing the internet and receiving traffic from anywhere.

c) Once External ALB receives the internet traffic it forwards it to the NGINX reverse proxy which have its own special configuration with instructions to forward all traffics received from the External ALB to the INTERNAL ALB

d) Internal ALB is expected to send this traffic to the webservers which in this case is WordPress server and the Tooling server. But how does it send it to the right one without making any mistake?

A rule has to be configured in our nginx reverse proxy server which would require the host header of

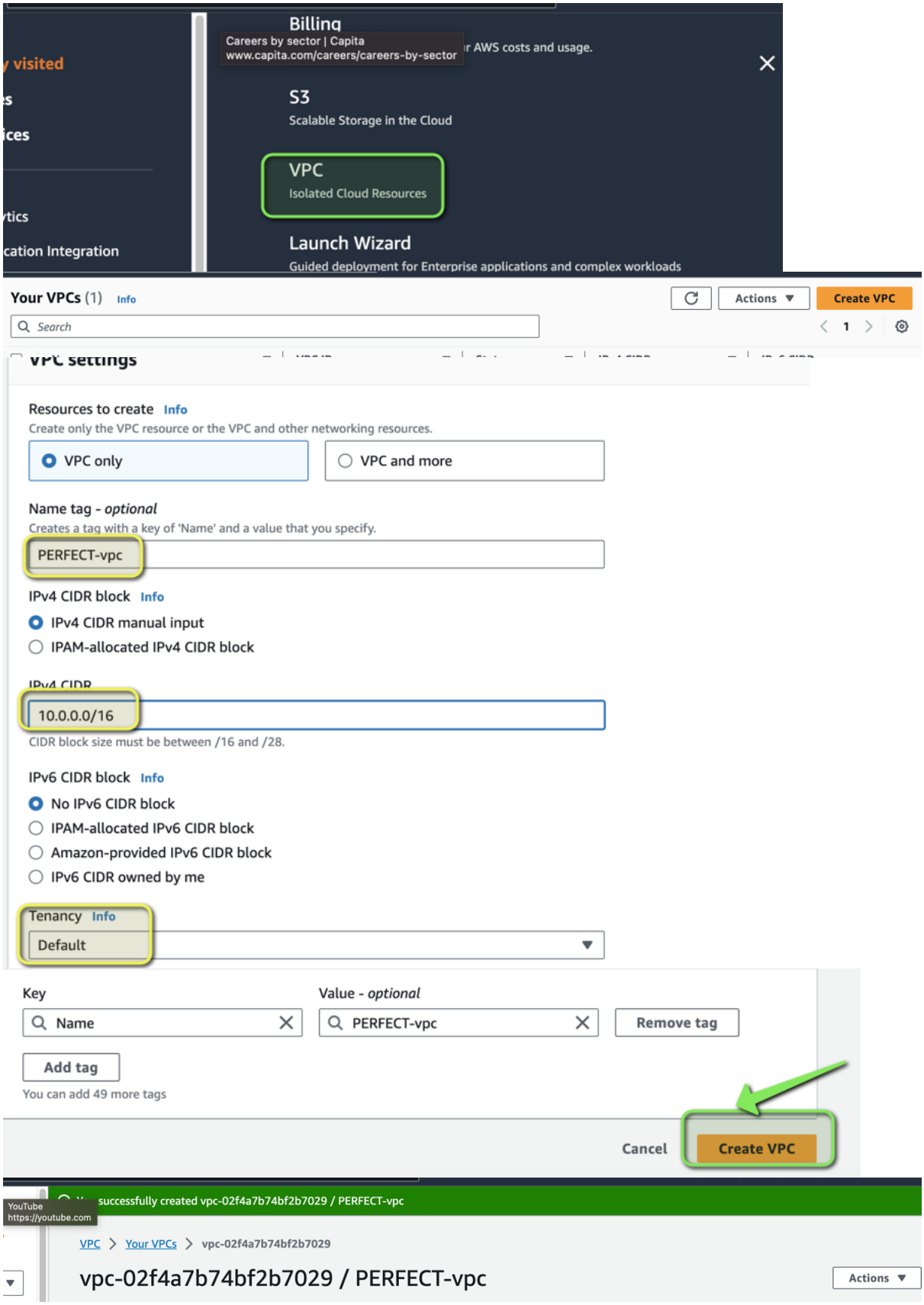
Default rule: Forward the host header of the webserver to the WordPress server via the Internal ALB

Conditional Rule: Check host header if “tooling” Route specific traffic to Internal ALB which in turn sends to the WordPress tooling server.

e) Our webservers would mount all their file system to the Elastic File System
f) The data would be stored in the RDS database
g) Bastion Server which is the entry point of all our resources but to effectively perform its role, the nginx concept needs to be considered .Nginx reverse proxy is the one that routes traffic from the External ALB downwards because of its access to the internet (public subnet). The Bastion needs to connect to the Nginx reverse proxy primarily to carry out administrative and management tasks which is the only way to connect to the private subnet, WordPress server as well as the WordPress tooling website .

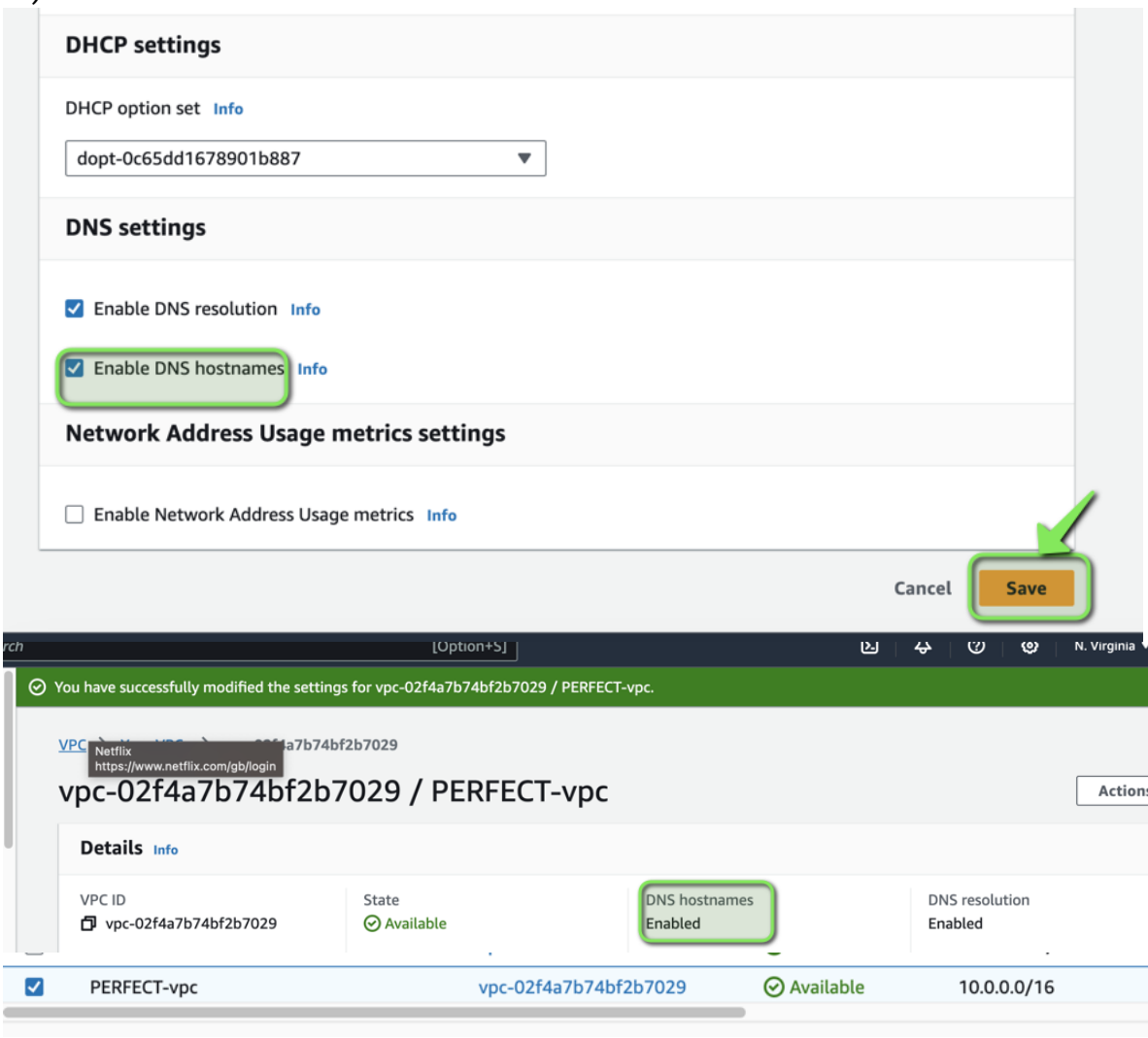
Its time to start creating all resources to kickstart the project. Please note that we are using 2 different availability zone to ensure high availability .

1) CREATE THE VPC:

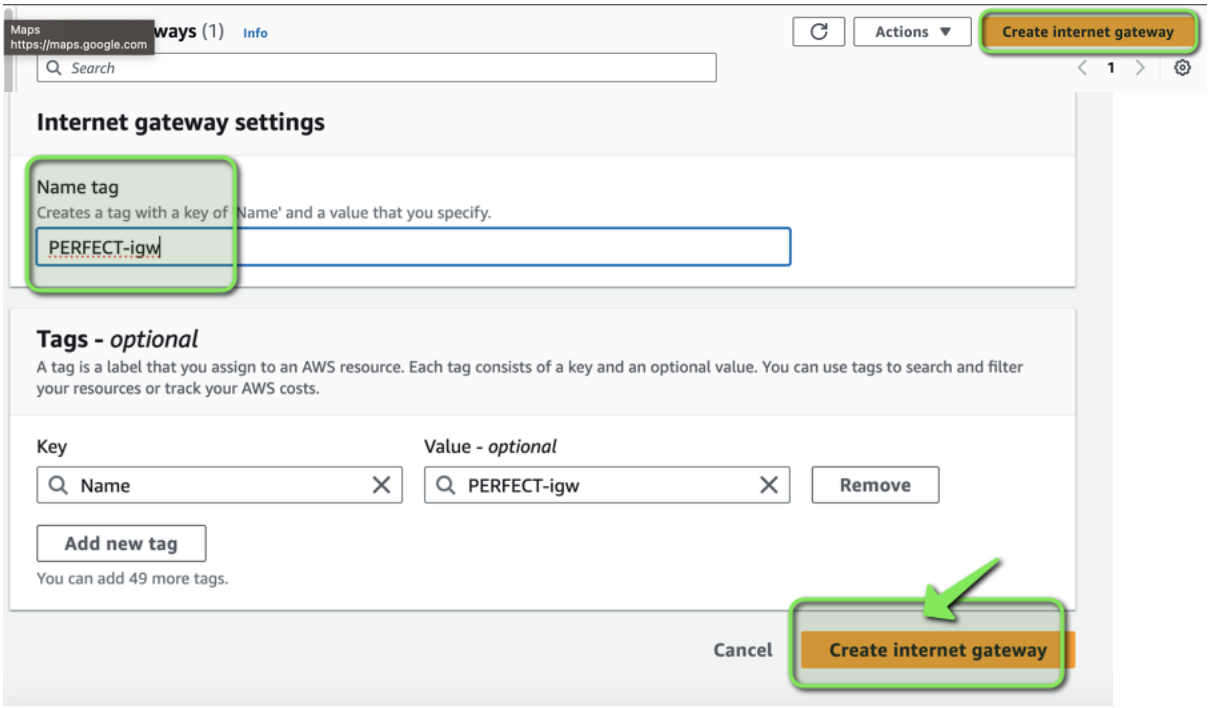

Billing
Careers by sector | Capita
www.capita.com/careers/careers-by-sector
r AWS costs and usage.
S3
Scalable Storage in the Cloud
VPC
Isolated Cloud Resources
Launch Wizard
Guided deployment for Enterprise applications and complex workloads
Your VPCs (1) Info
Search
Actions
Create VPC
1
Resources to create Info
Create only the VPC resource or the VPC and other networking resources.
VPC only
VPC and more
Name tag - optional
Creates a tag with a key of 'Name' and a value that you specify.
PERFECT-vpc
IPv4 CIDR block Info
IPv4 CIDR manual input
IPAM-allocated IPv4 CIDR block
10.0.0.0/16
CIDR block size must be between /16 and /28.
IPv6 CIDR block Info
No IPv6 CIDR block
IPAM-allocated IPv6 CIDR block
Amazon-provided IPv6 CIDR block
IPv6 CIDR owned by me
Tenancy Info
Default
Key
Value - optional
Name
PERFECT-vpc
Remove tag
Add tag
You can add 49 more tags
Cancel
Create VPC
successfully created vpc-02f4a7b74bf2b7029 / PERFECT-vpc
VPC > Your VPCs > vpc-02f4a7b74bf2b7029
vpc-02f4a7b74bf2b7029 / PERFECT-vpc
Actions

VPC has been created .

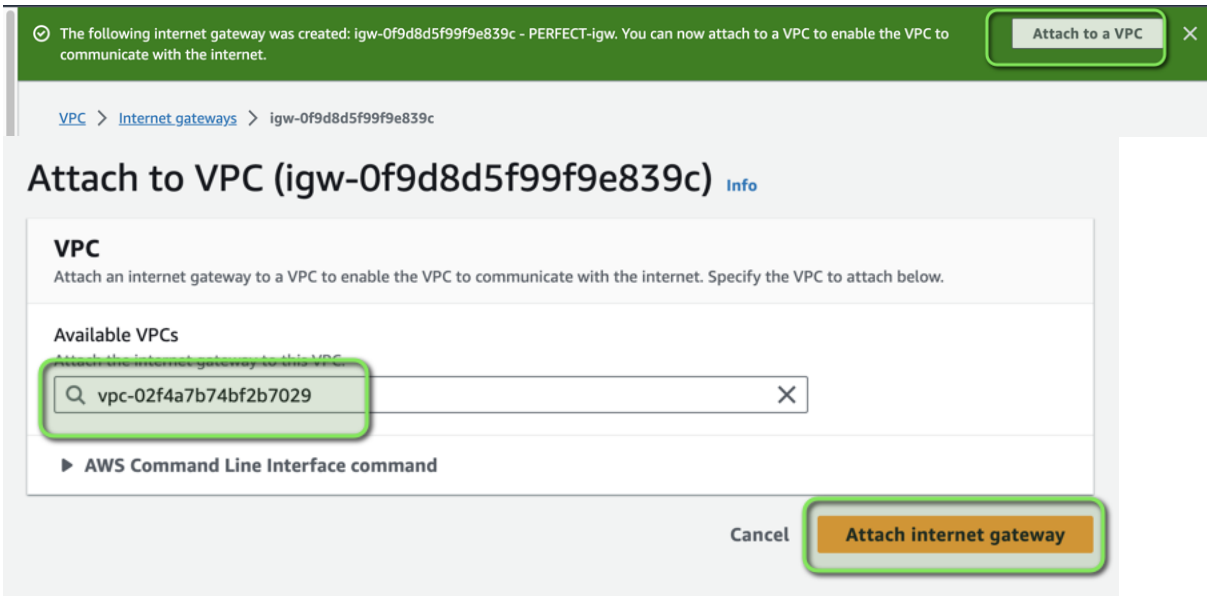
2) Edit and enable DNS Hostname



3) Create an Internet Gateway



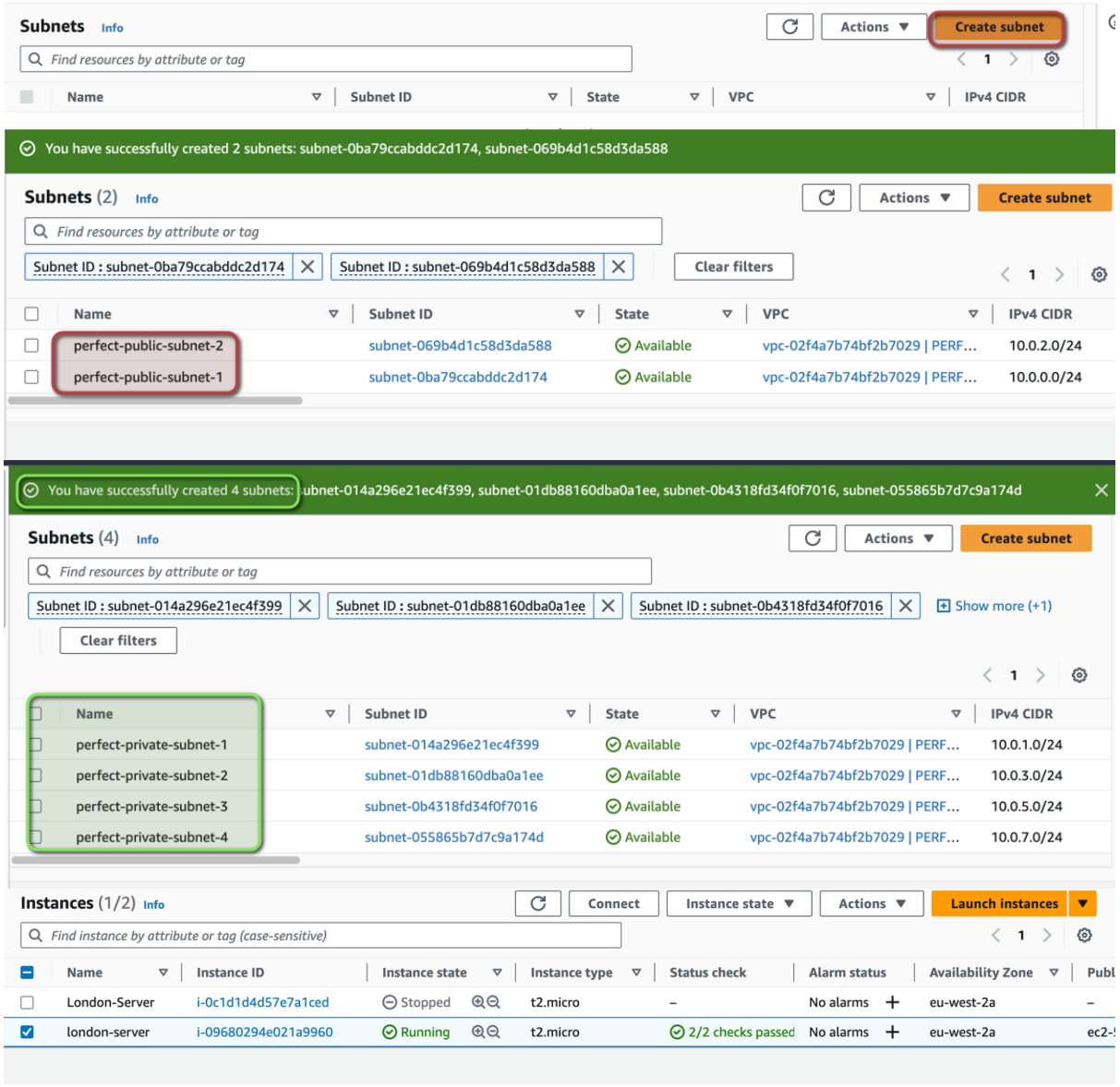
4) Attach Internet Gateway to the VPC



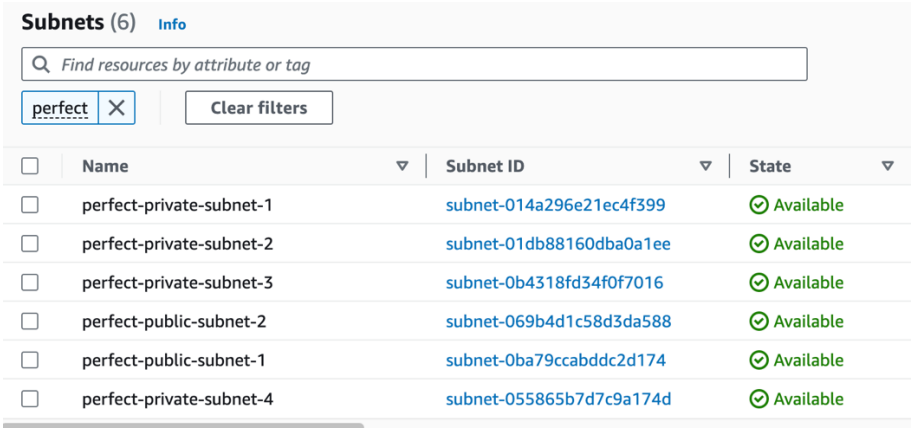
Attached successfully.

5) Create Private and Public Subnets.

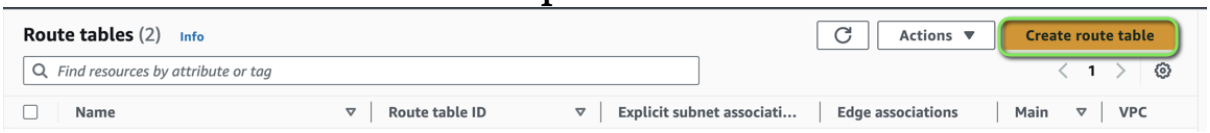
Please Note: All public subnets are even numbers while private subnets are odd numbers

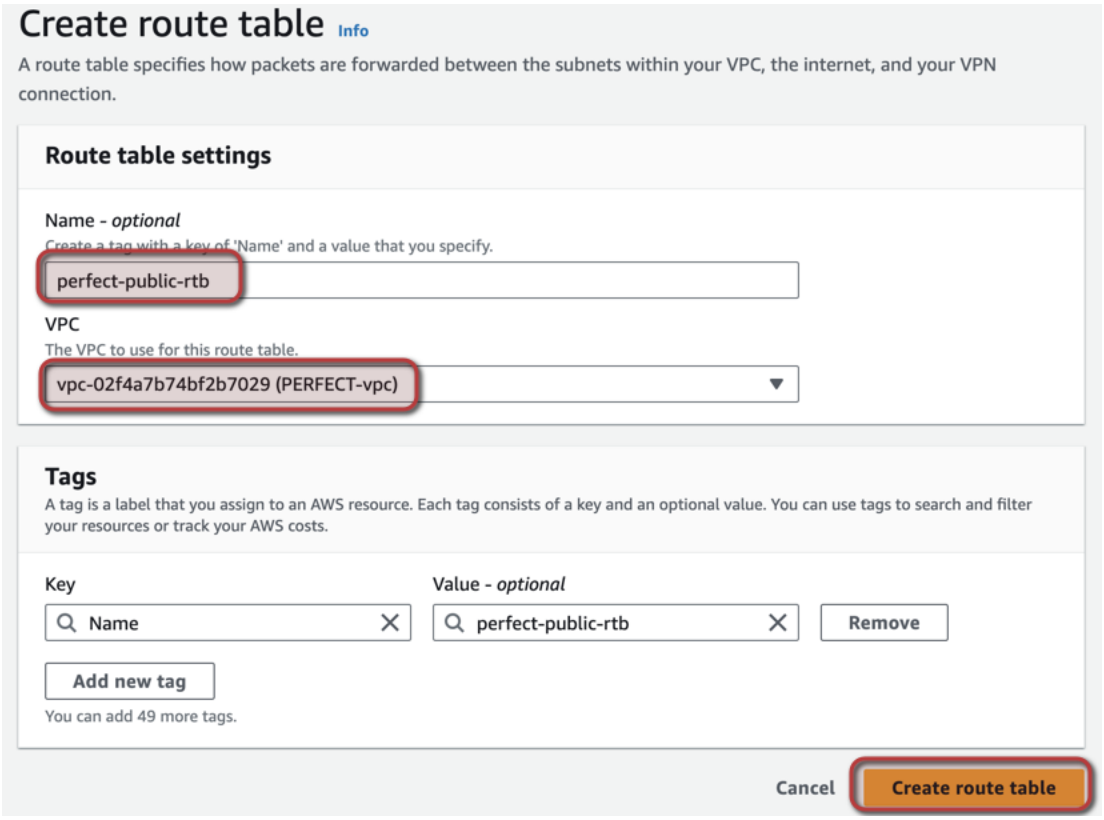


All subnets are created .

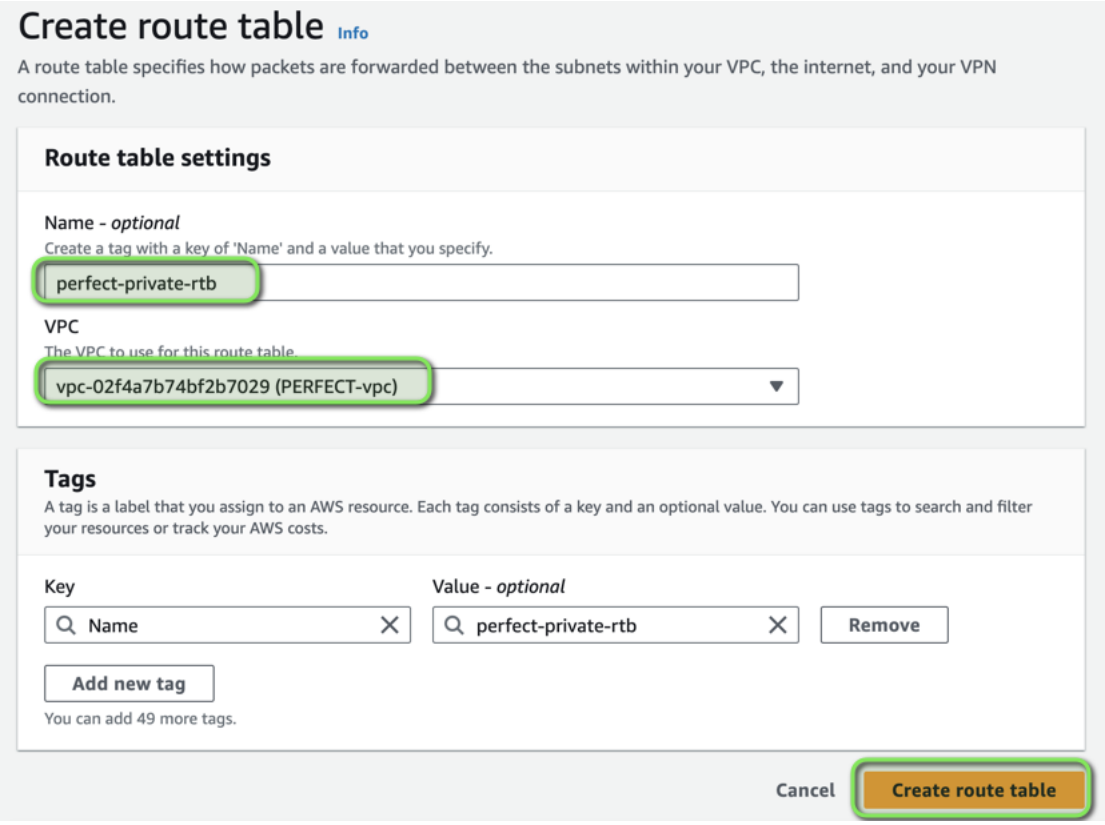


6) Create route table for the public subnet :

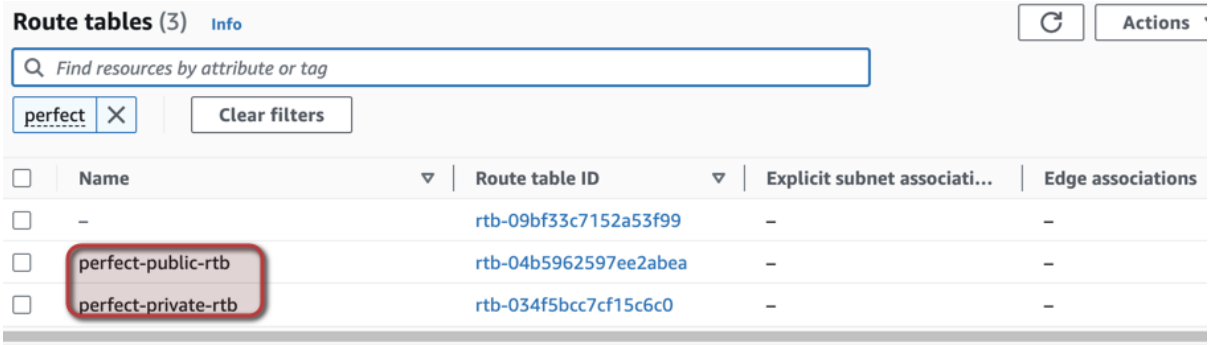




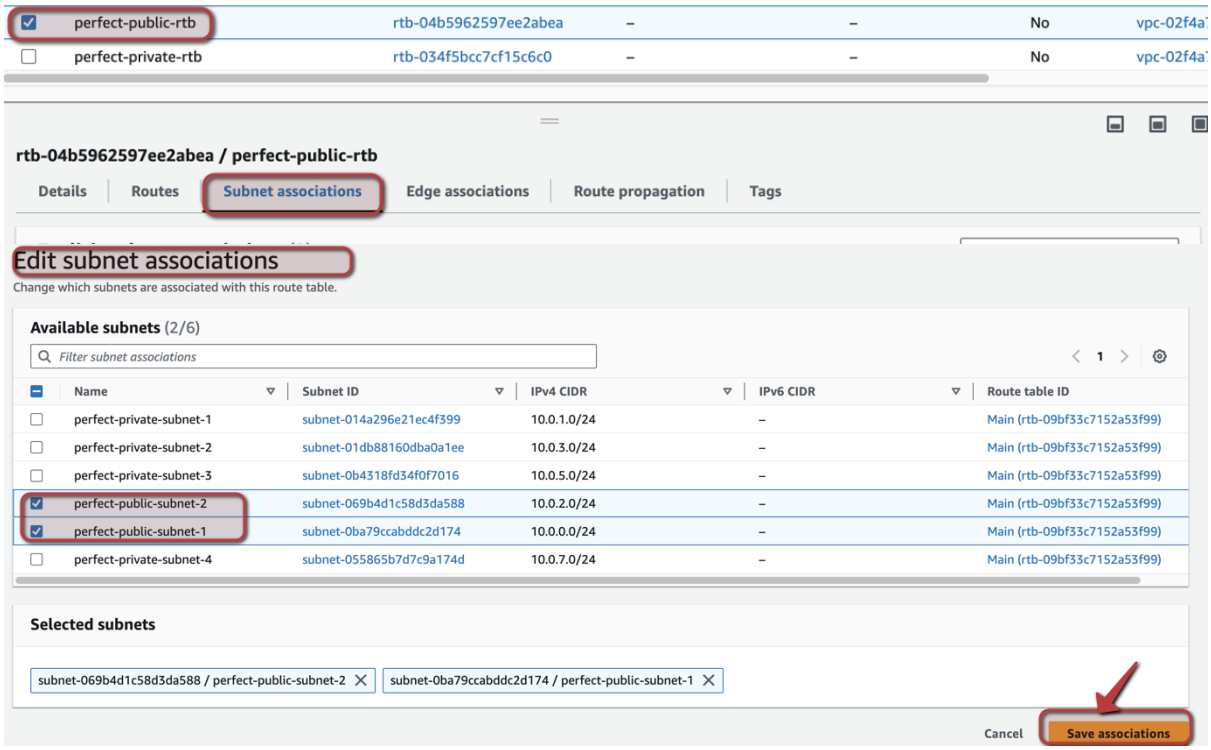
7) Create route table for the private subnet :



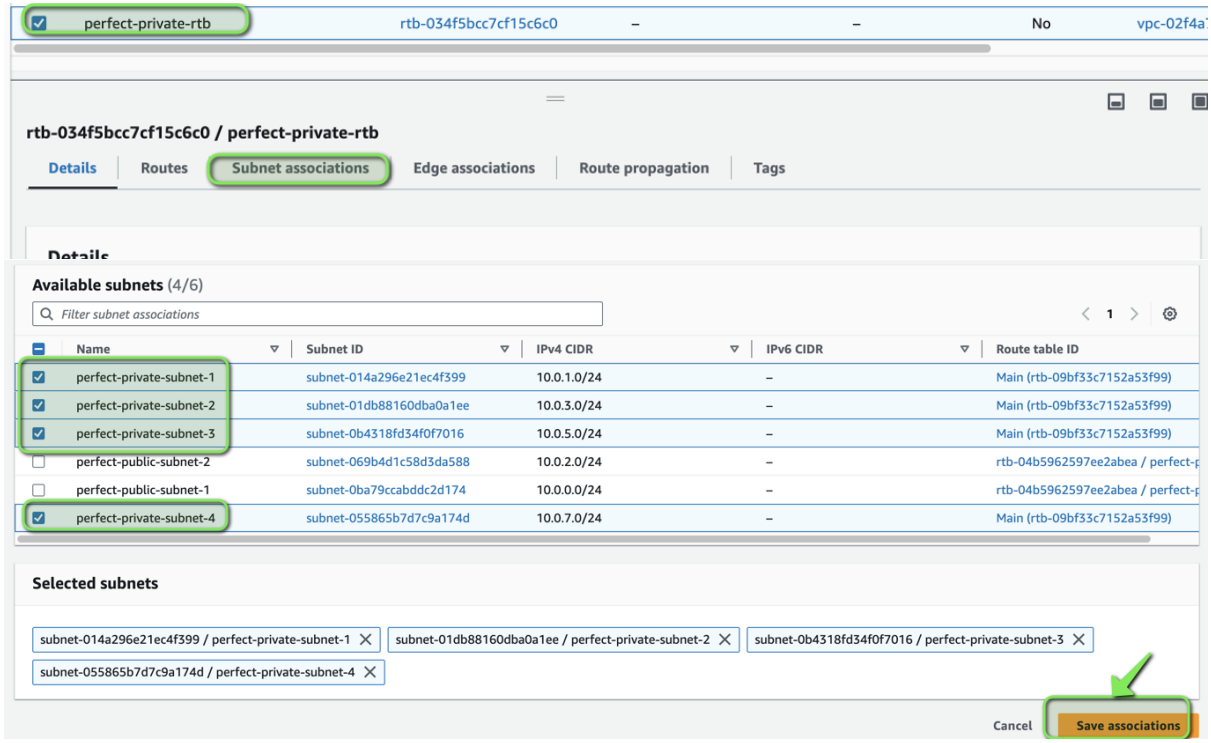
Route tables created



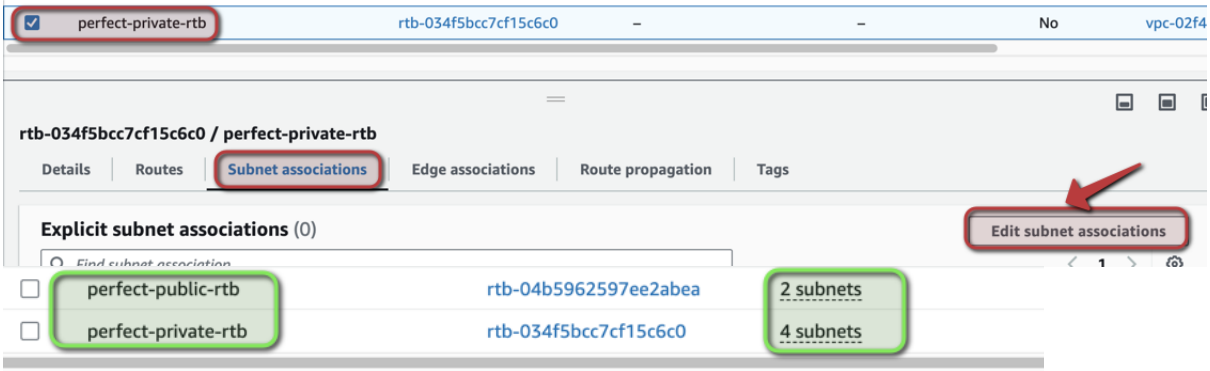
8) Perform subnet association to your created route tables.



Public route table associated.

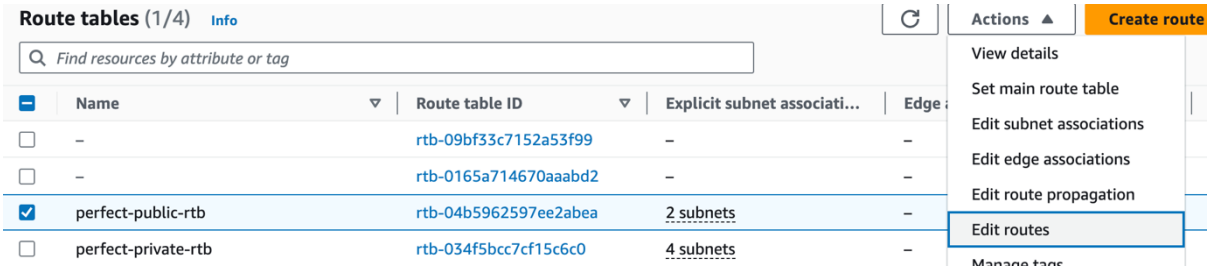


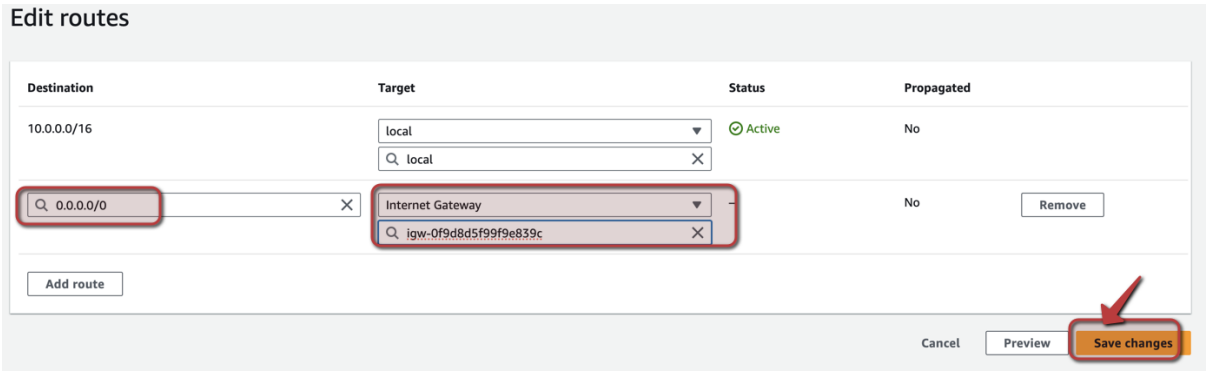
Private route table associated



All route table associated.

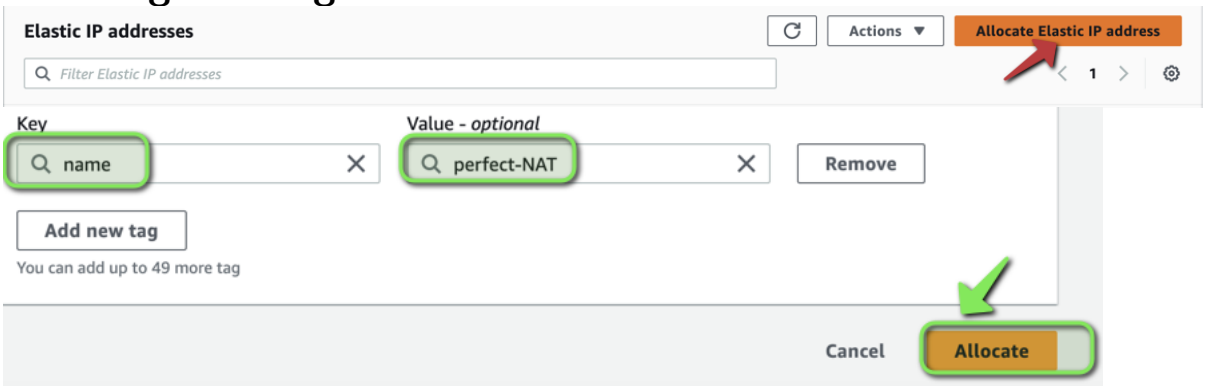
9) Editing route of the route table for public subnet to have access to the internet through the internet gateway. This would allow our public subnet be accessible to the internet .



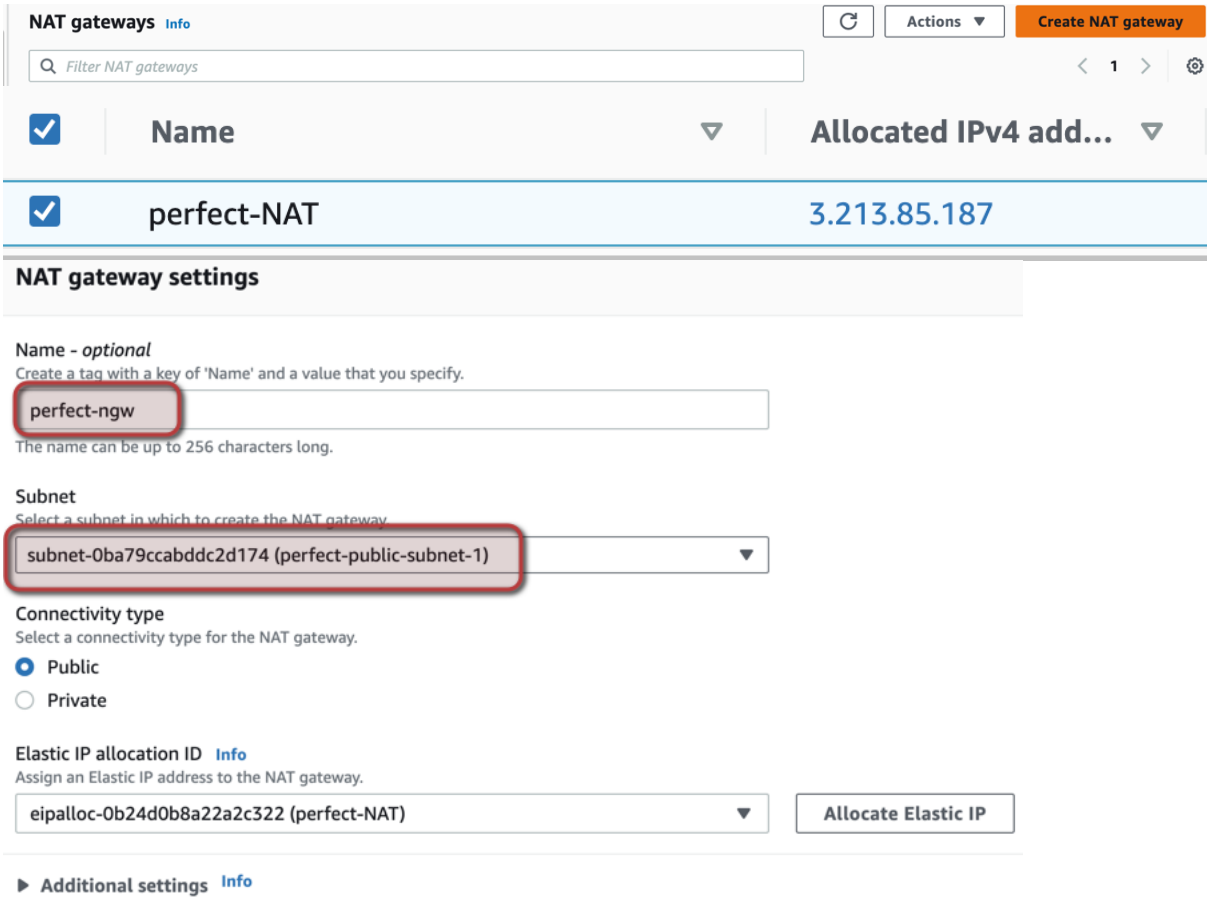


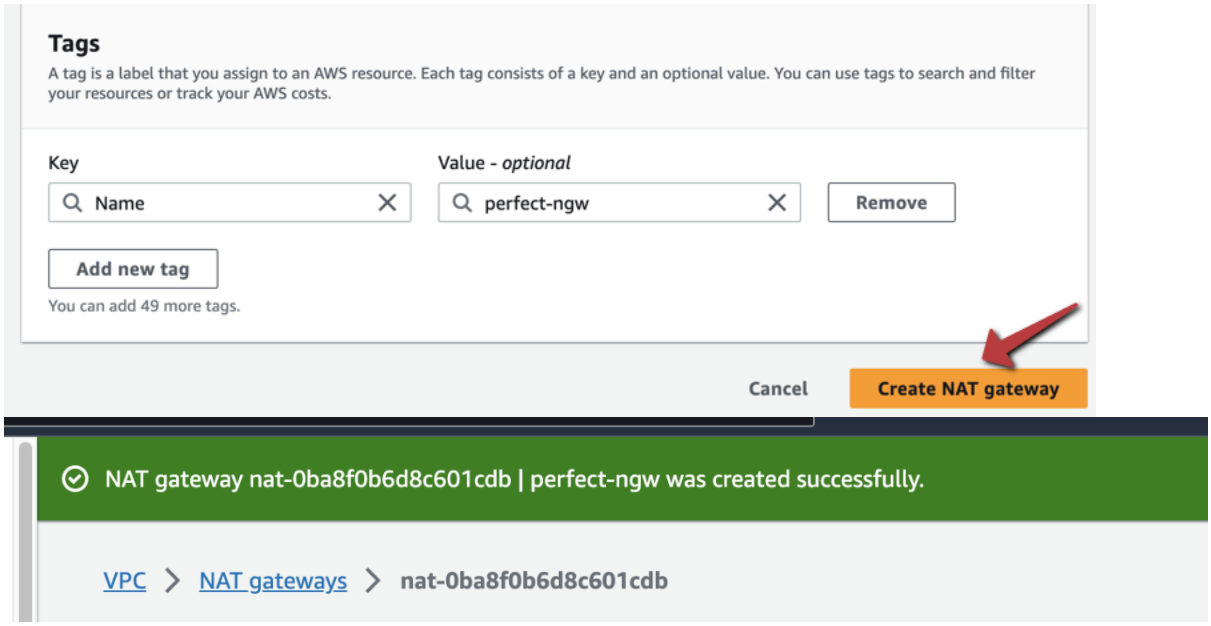
10) Create an Elastic IP address (stable IP ADDRESS) that does not change .This would be needed by the NAT GATEWAY which is needed for private subnet access .

i) Elastic IP address allocated .Please note all Elastic IP must be allocated and you wont get charged but if you do not allocate it you would get charged .



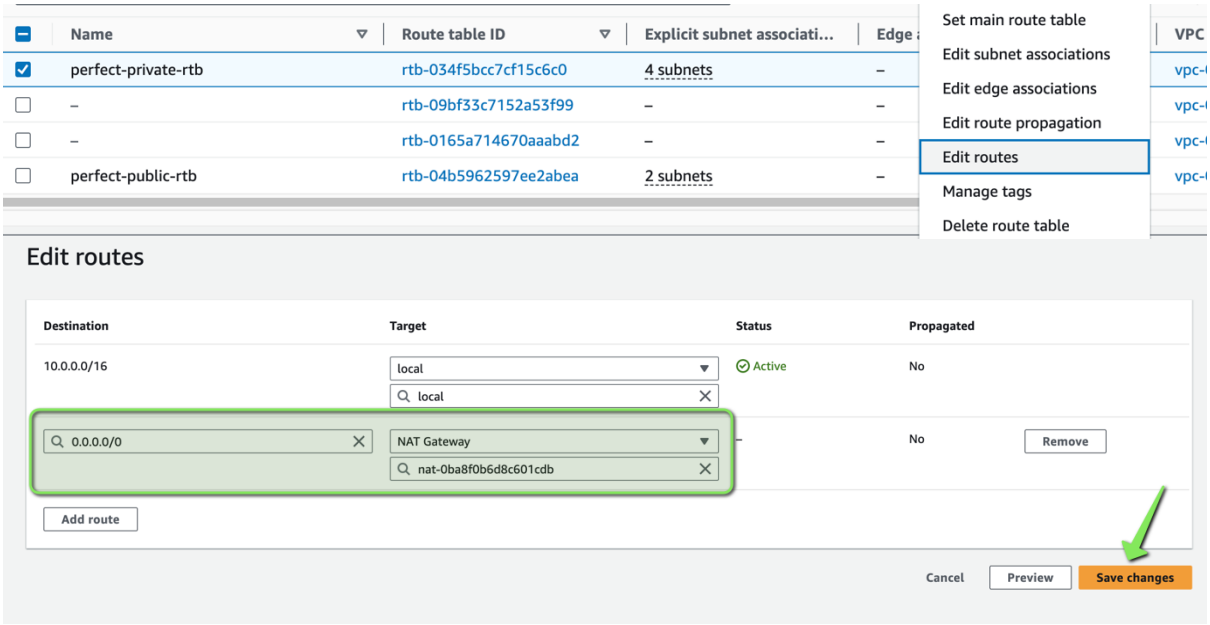
11)Create NATGATEWAY. Please note they are always created on a public subnet 1 as shown below .





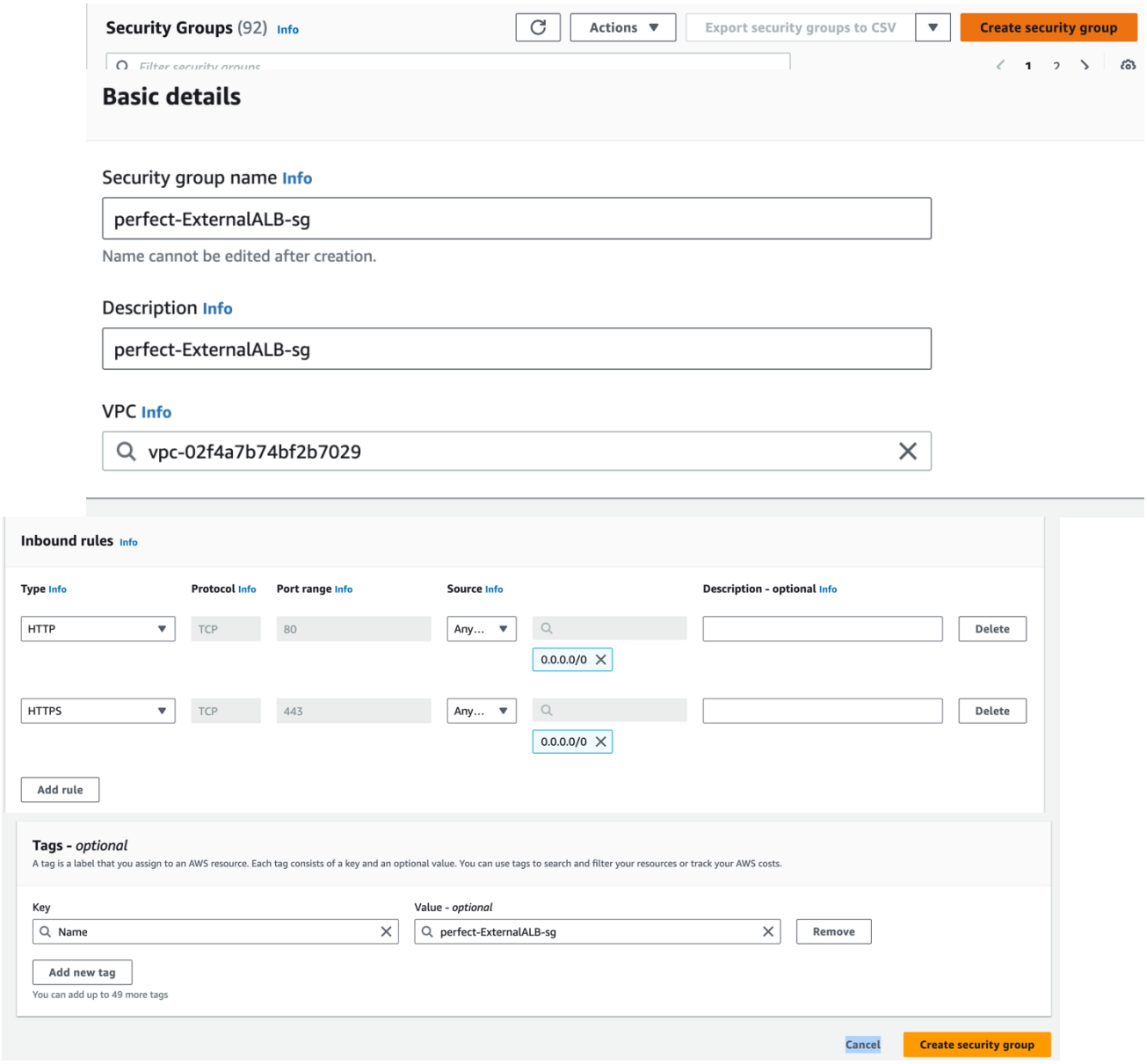
NAT GATEWAY created successfully .

11) Navigate to the private route table and edit route.

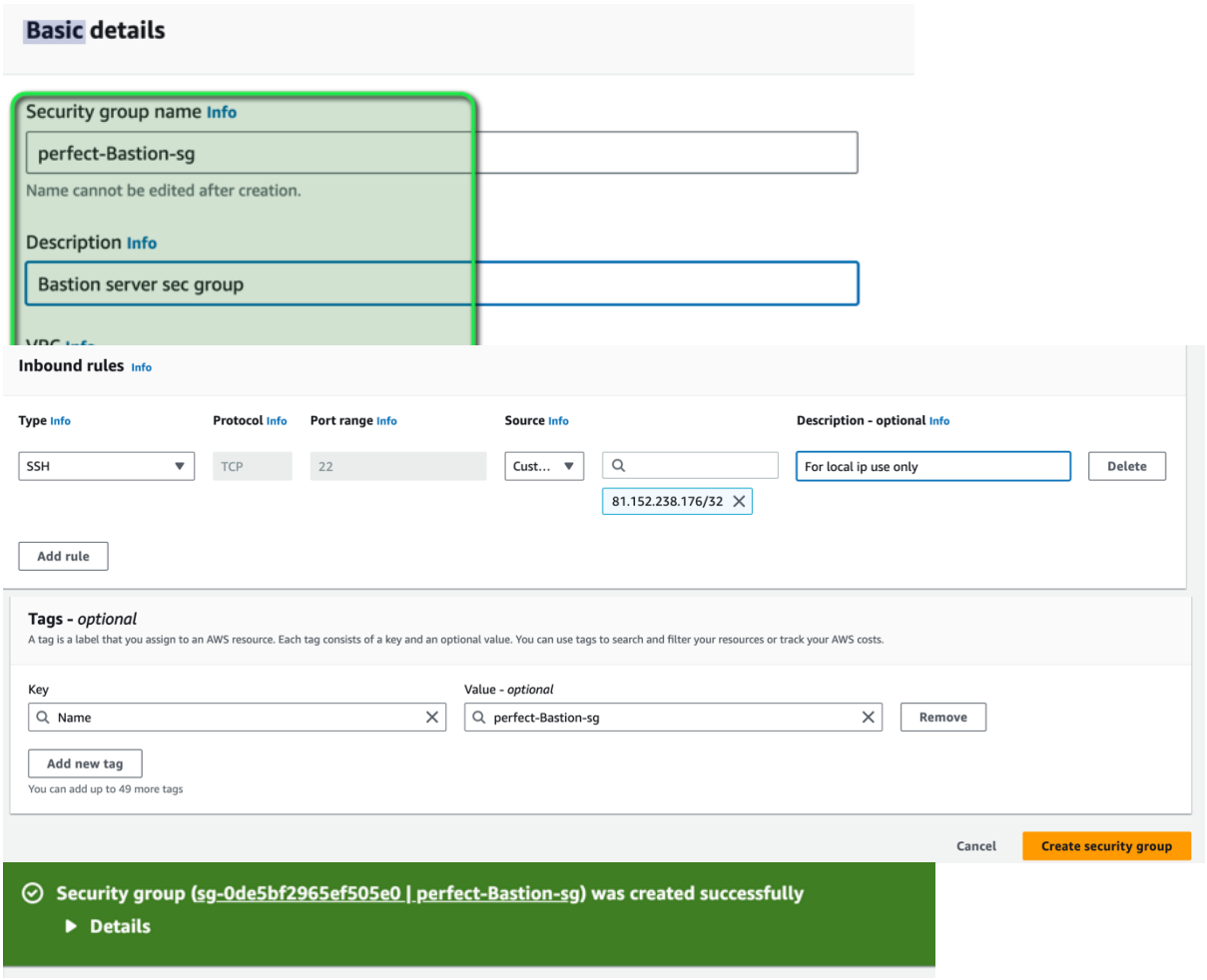


We have configured our route table and then we proceed to create security groups for other resources .

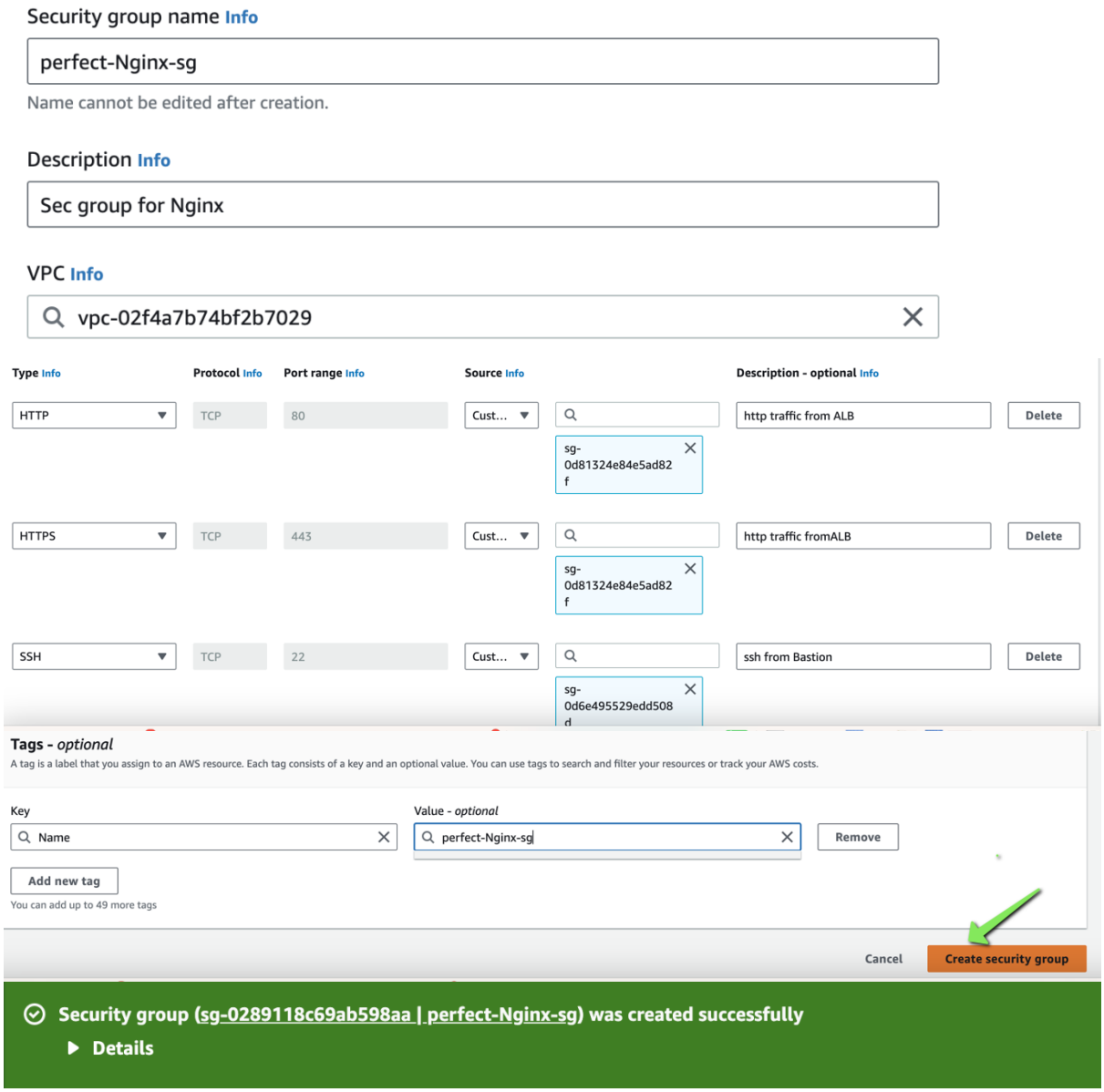
12) Create security group for external ALB. Open traffic for (HTTP AND HTTPS) for inbound rules .



13) Create security group for Bastion . Open traffic for (SSH traffic only to our computer) for inbound rules .



14) Create security group for Nginx. Open traffic for (SSH from Bastion , HTTP AND HTTPS from External ALB) for inbound rules .



15) Create security group for internal ALB. Open traffic for (HTTP AND HTTPS) for inbound rules.

Basic details

Security group name Info

perfect-InternalALB-sg

Name cannot be edited after creation.

Description Info

Sec group for perfect-InternalALB

VPC Info

vpc-02f4a7b74bf2b7029

Inbound rules Info

Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
HTTP	TCP	80	Any...	0.0.0.0/0	HTTP access for Internal ALB HTTP access for Internal ALB	Delete
HTTPS	TCP	443	Any...	0.0.0.0/0	HTTPS access for Internal ALB HTTPS access for Internal ALB	Delete

Add rule

Tags - *optional*

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value - *optional*	
Name	perfect-InternalALB-sg	Remove

Add new tag

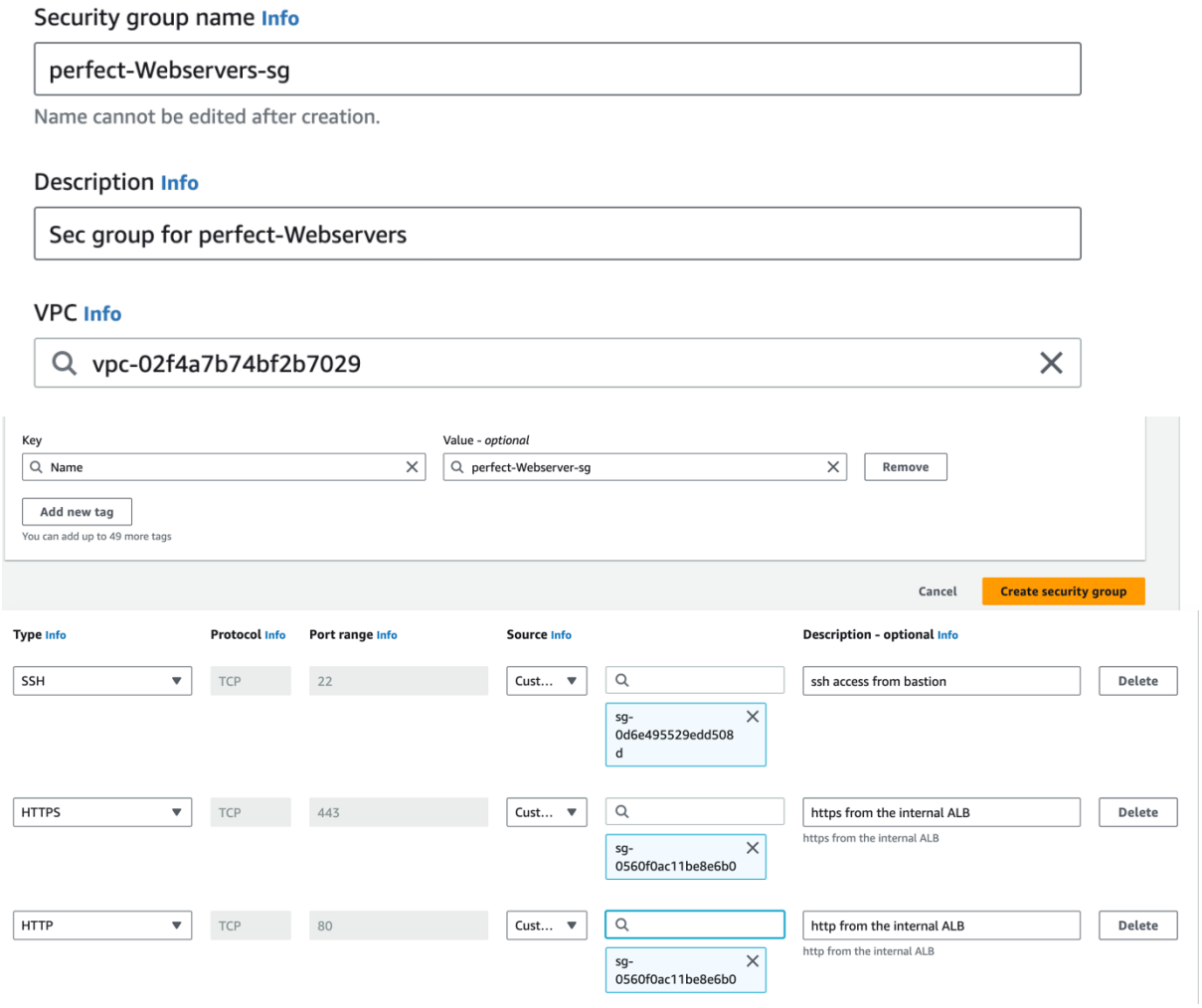
You can add up to 49 more tags

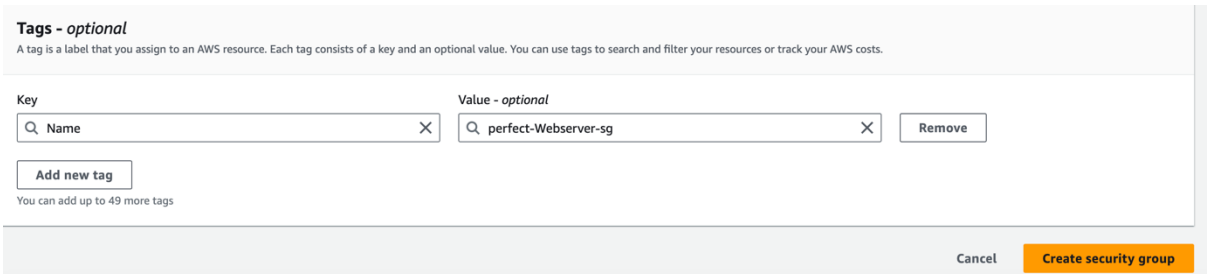
Cancel | Create security group

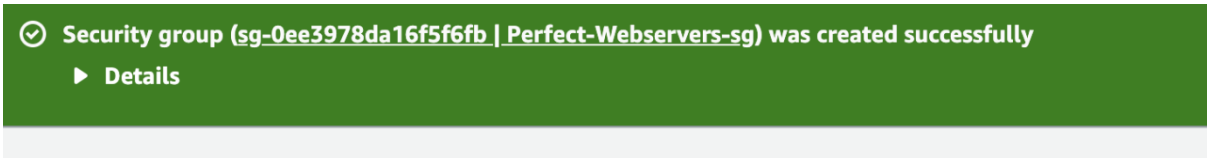
Security group (sg-07fda2e31f7b82e77 | perfect-InternalALB-sg) was created successfully

▶ **Details**

16) Create security group for Webserver. Open traffic for (SSH from Bastion , HTTP AND HTTPS) for inbound rules .





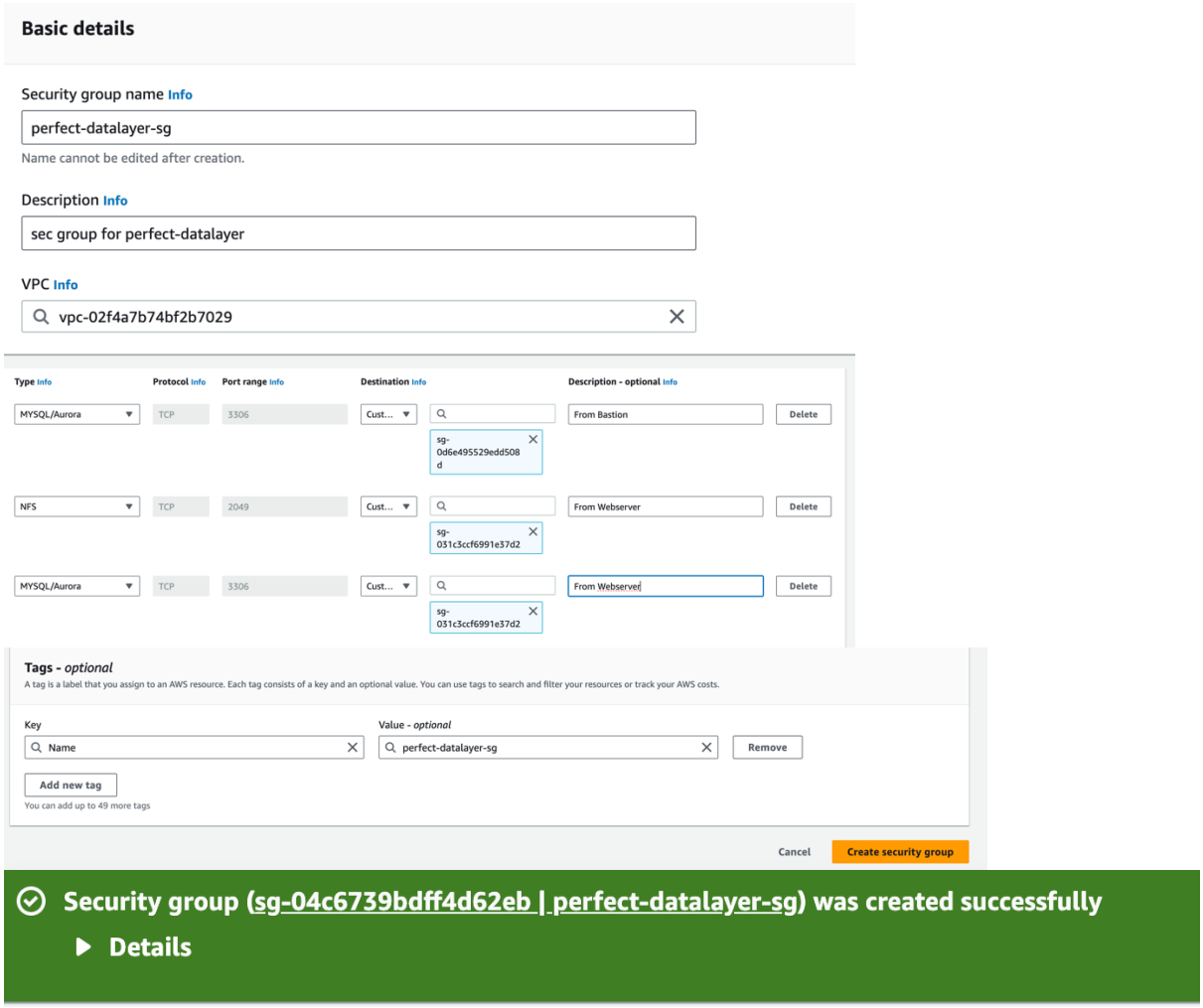


16) Create security group for Data Layer. Our data layer has the Amazon EFS and RDS

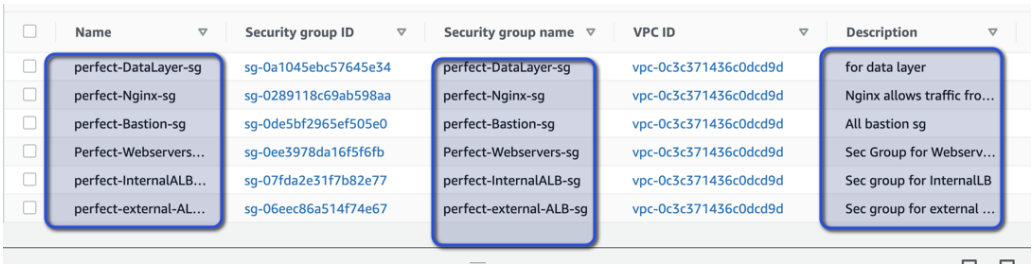
a) Our webserver needs to mount the file system and needs an NFS access.

b) Connect the webserver to the database (MySQL Access)

c) Bastion needs administrative access to the data base (MySQL Access)



All security groups are completed as shown below.

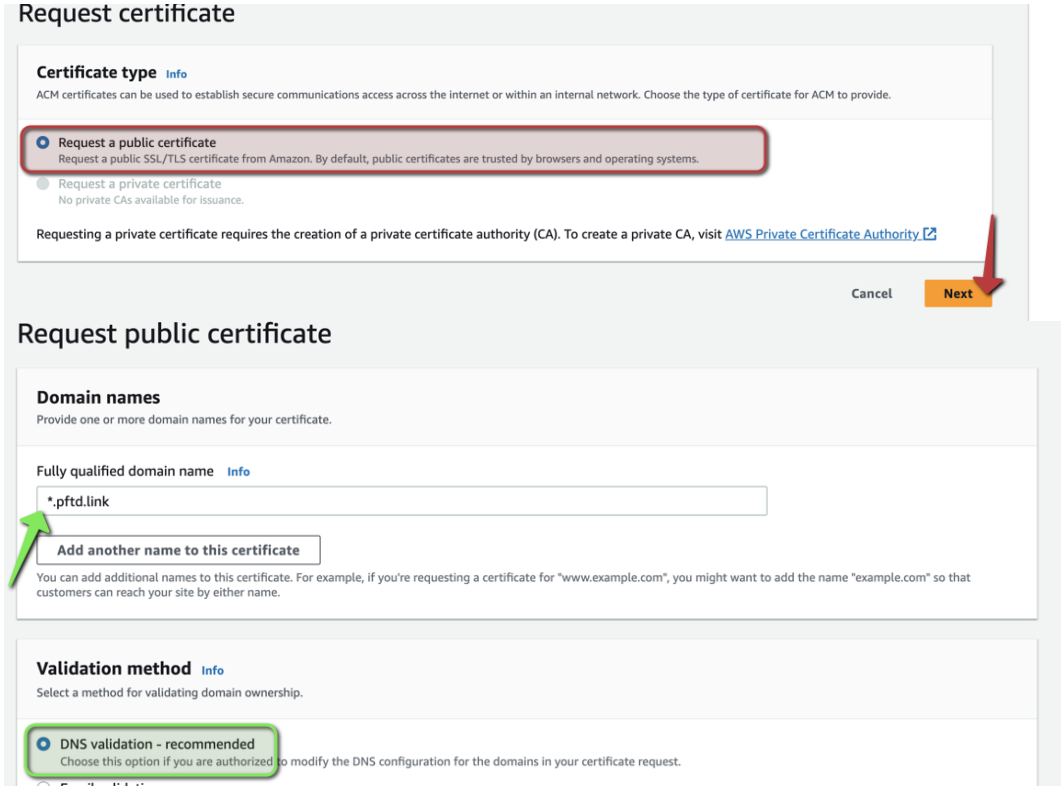


Name	Security group ID	Security group name	VPC ID	Description
perfect-DataLayer-sg	sg-0a1045ebc57645e34	perfect-DataLayer-sg	vpc-0c3c371436c0dcd9d	for data layer
perfect-Nginx-sg	sg-0289118c69ab598aa	perfect-Nginx-sg	vpc-0c3c371436c0dcd9d	Nginx allows traffic fro...
perfect-Bastion-sg	sg-0de5bf2965ef505e0	perfect-Bastion-sg	vpc-0c3c371436c0dcd9d	All bastion sg
Perfect-Webservers...	sg-0ee3978da16f5f6fb	Perfect-Webservers-sg	vpc-0c3c371436c0dcd9d	Sec Group for Webserv...
perfect-InternalALB...	sg-07fda2e31f7b82e77	perfect-InternalALB-sg	vpc-0c3c371436c0dcd9d	Sec group for InternalLB
perfect-external-AL...	sg-06eec86a514f74e67	perfect-external-ALB-sg	vpc-0c3c371436c0dcd9d	Sec group for external ...

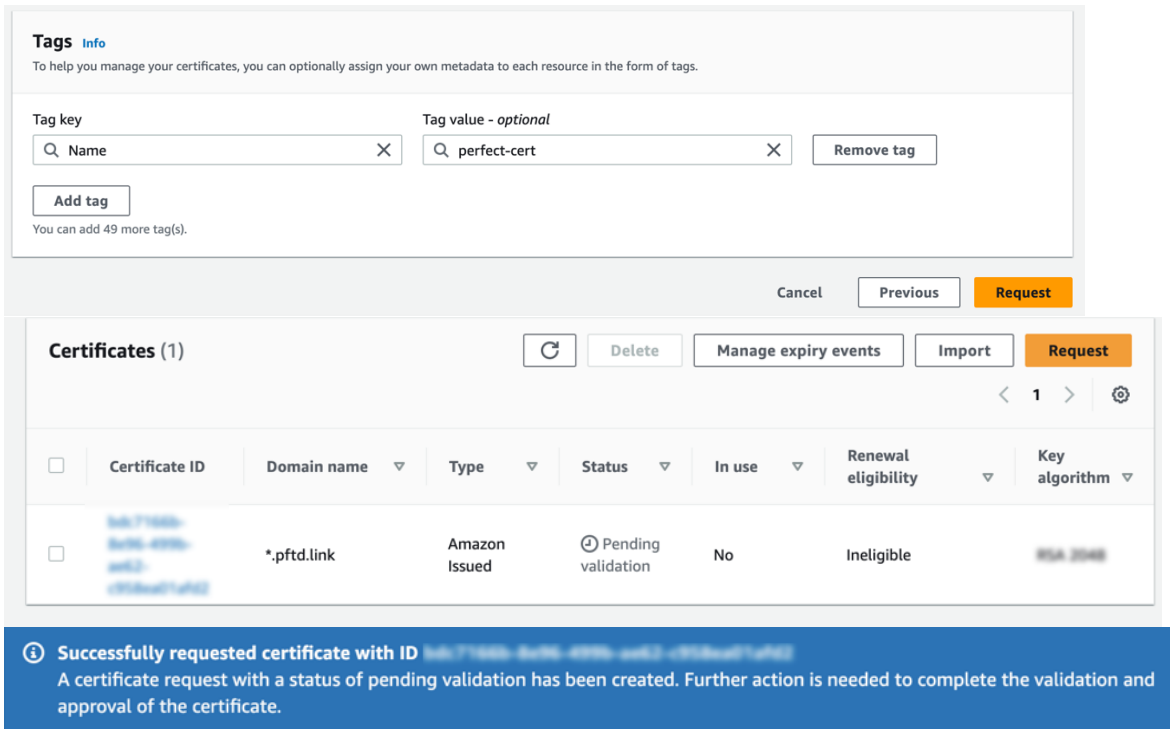
17)The next steps would be to create a new domain name and register it putting it on route53 DNS.

Domain name created: “ptfd.link”

Request for a certificate:

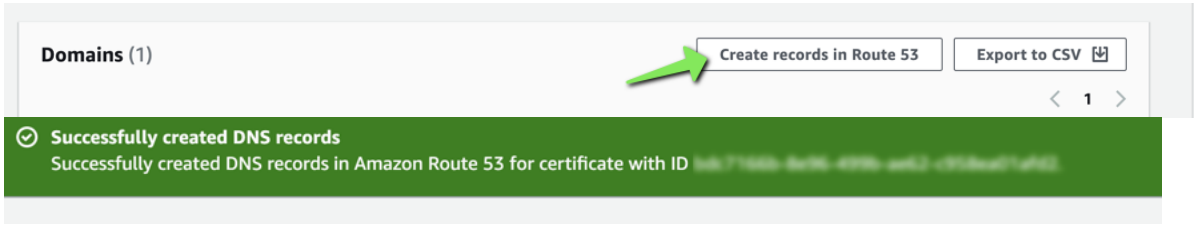


Please note We put an asterisk * to let our domain name have a wider range of choices and not restrict it.

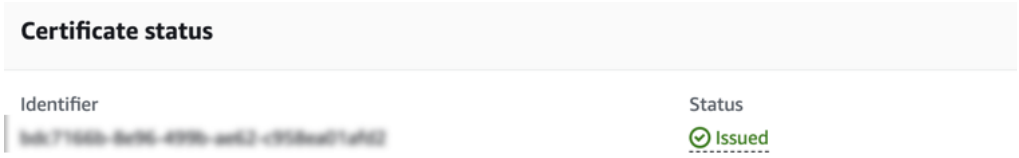


Certificate requested successfully but pending validation.

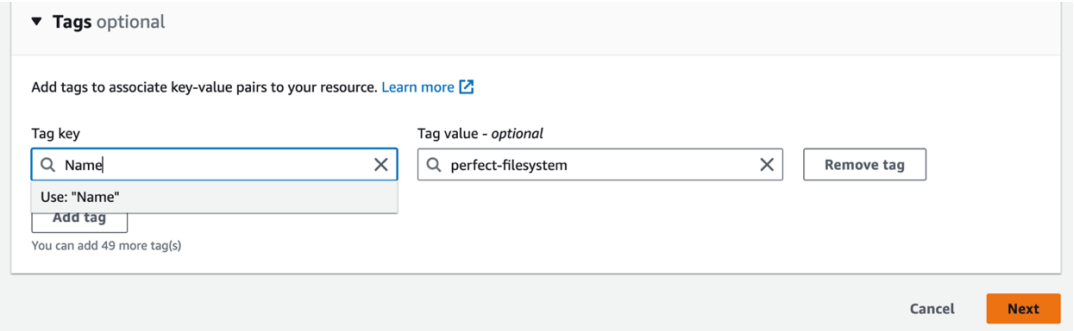
We create the records in Route DNS.



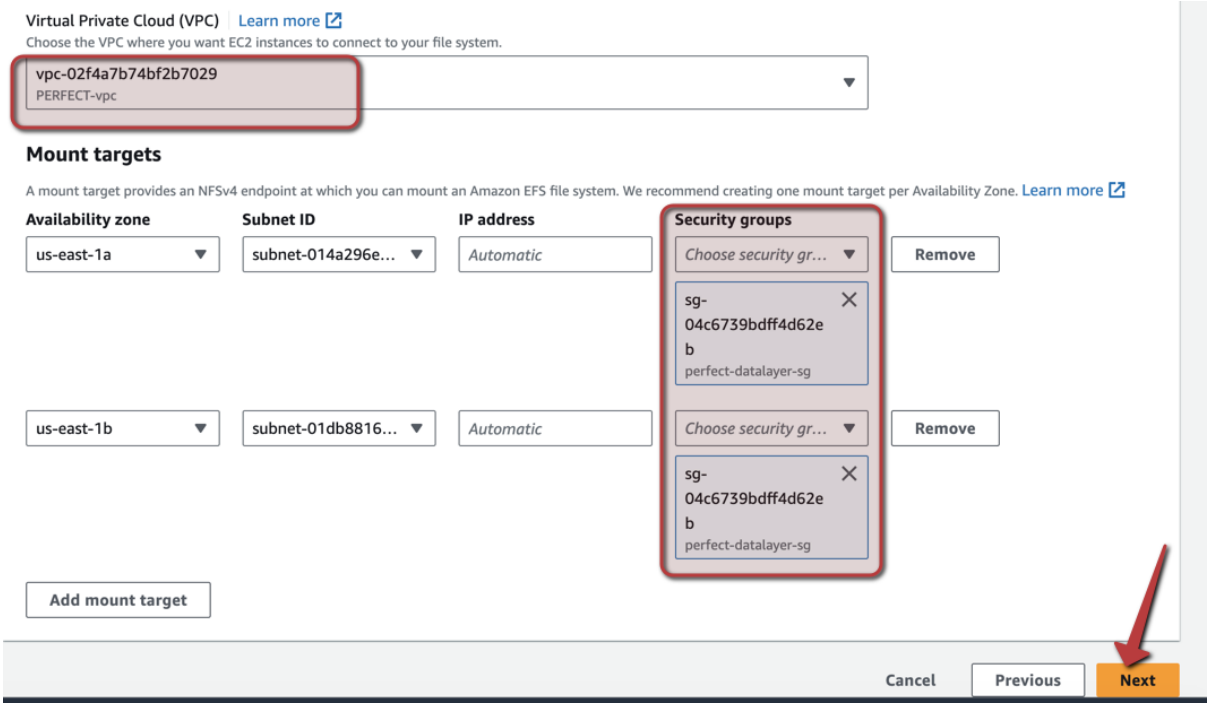
Once created ,on refresh ,Certificate issued below.



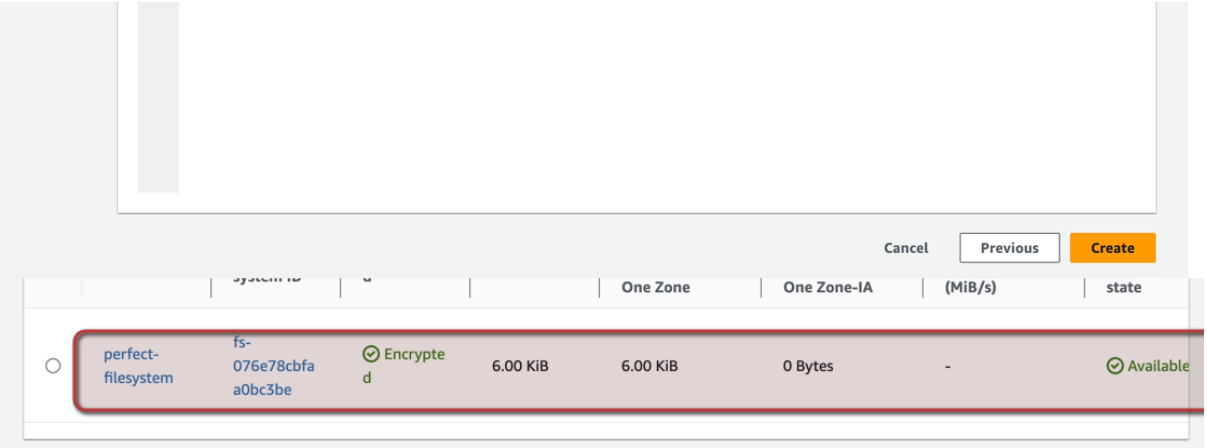
18) We then proceed to create and customise our Amazon Elastic File System.



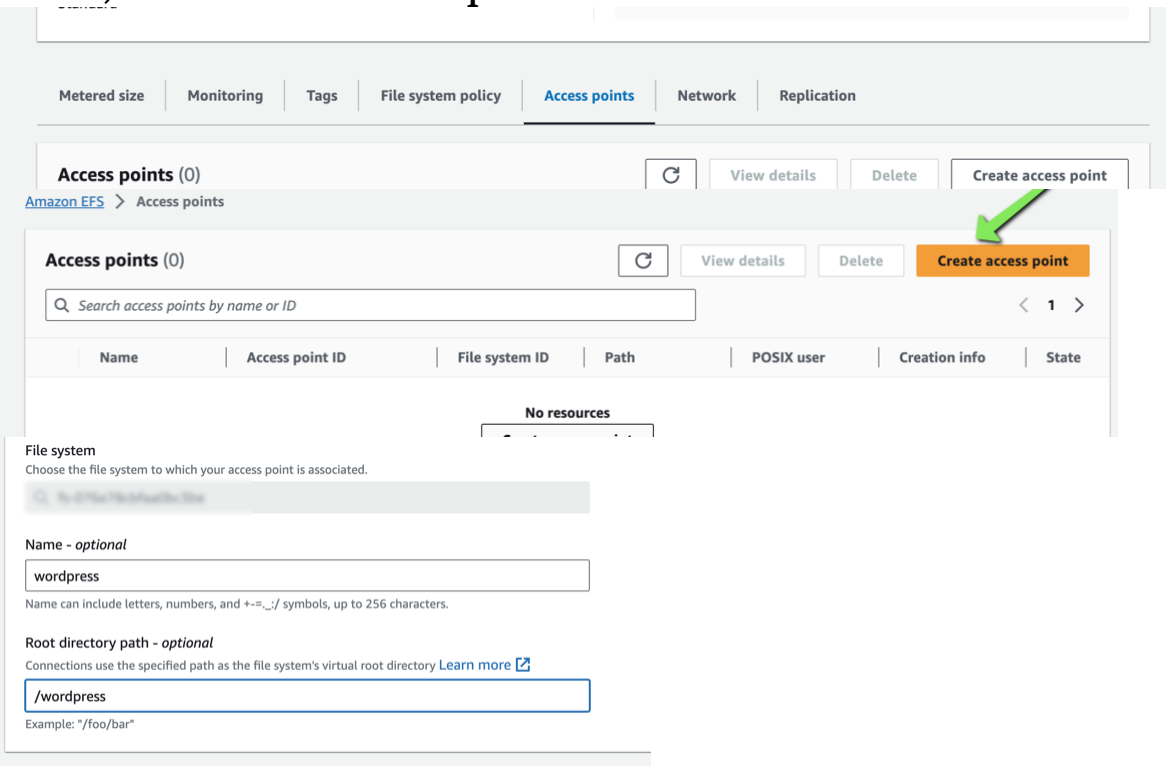
Specify the private subnets by routing into the file system as our webservers exists there. Once done the EFS becomes available to the webservers.
Fill in the network access .(Mount targets to the file system)

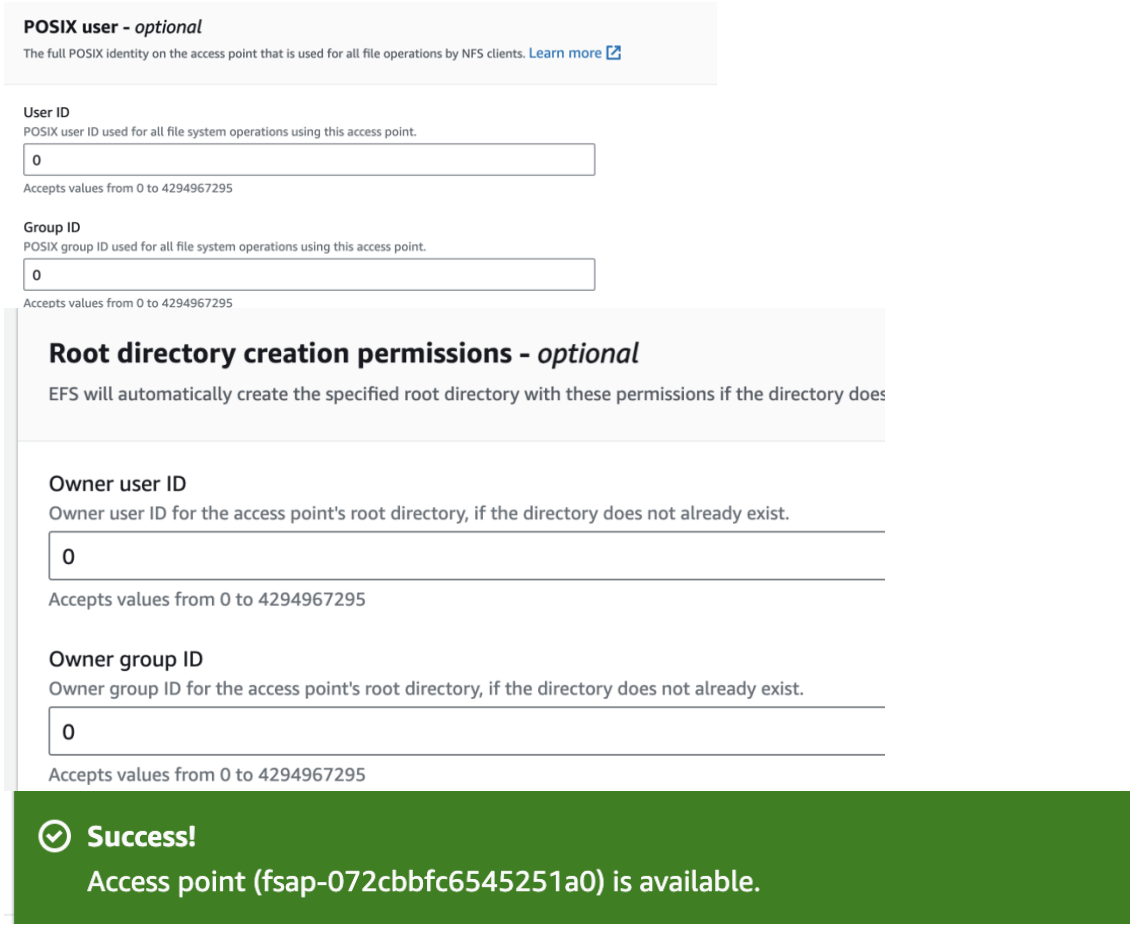


Click Next and Create the EFS. It is also important to note that EFS file system is created and encrypted.

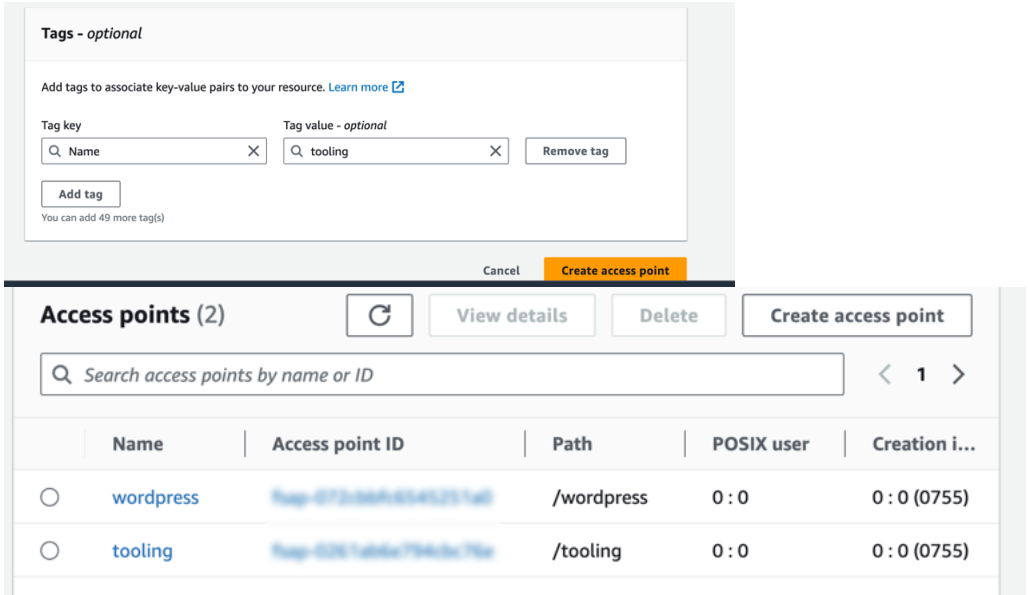


19)Create an access point for WordPress webserver.



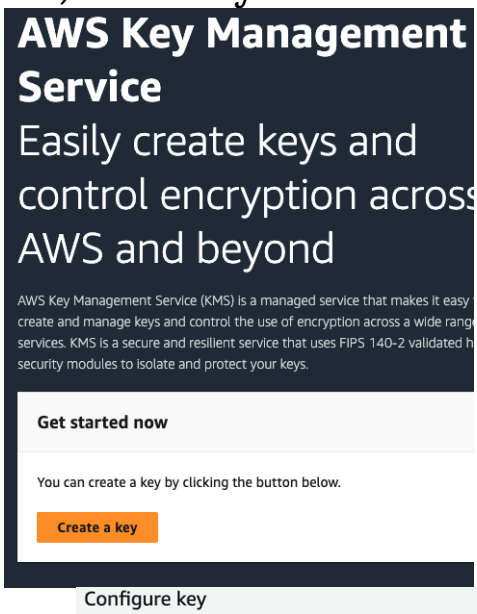


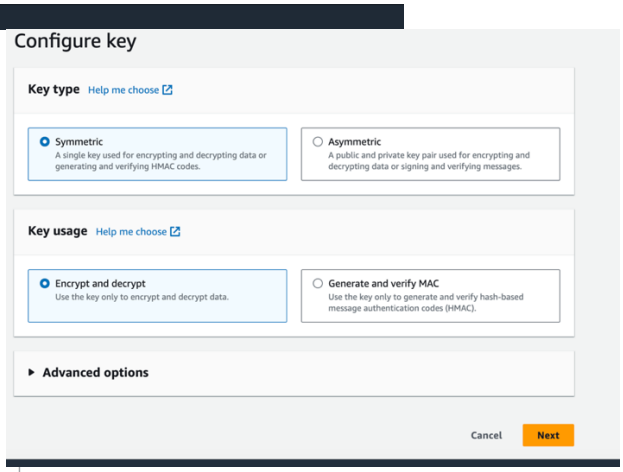
20) Create an access point for tooling

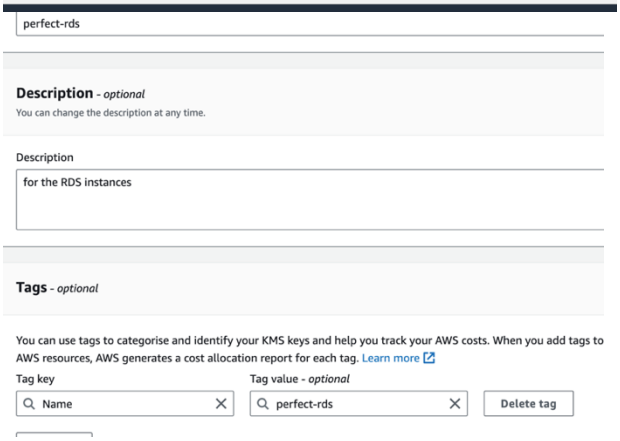


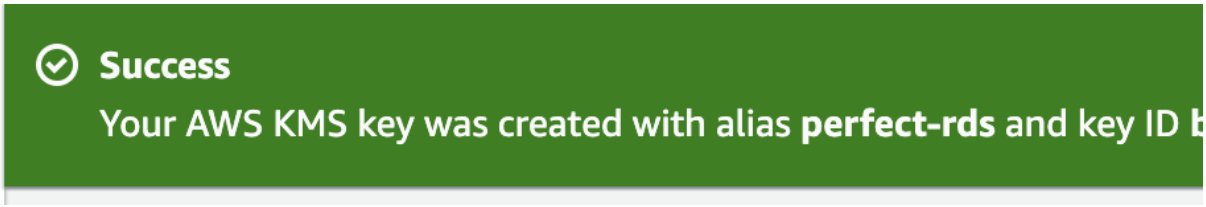
We have completed our EFS creation. Next would be creating our RDS but we need to create a KMS key to be used to encrypt the database instance .

21)KMS Key and RDS instance creation



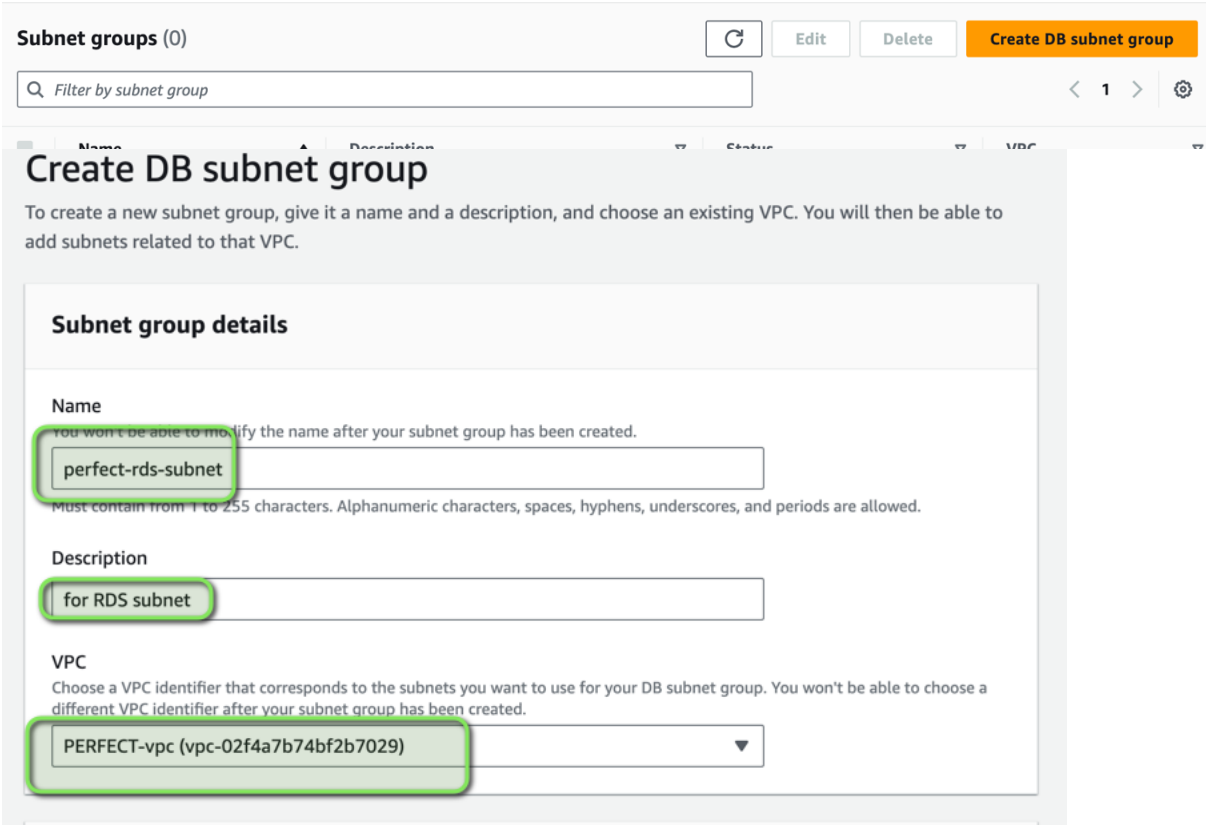


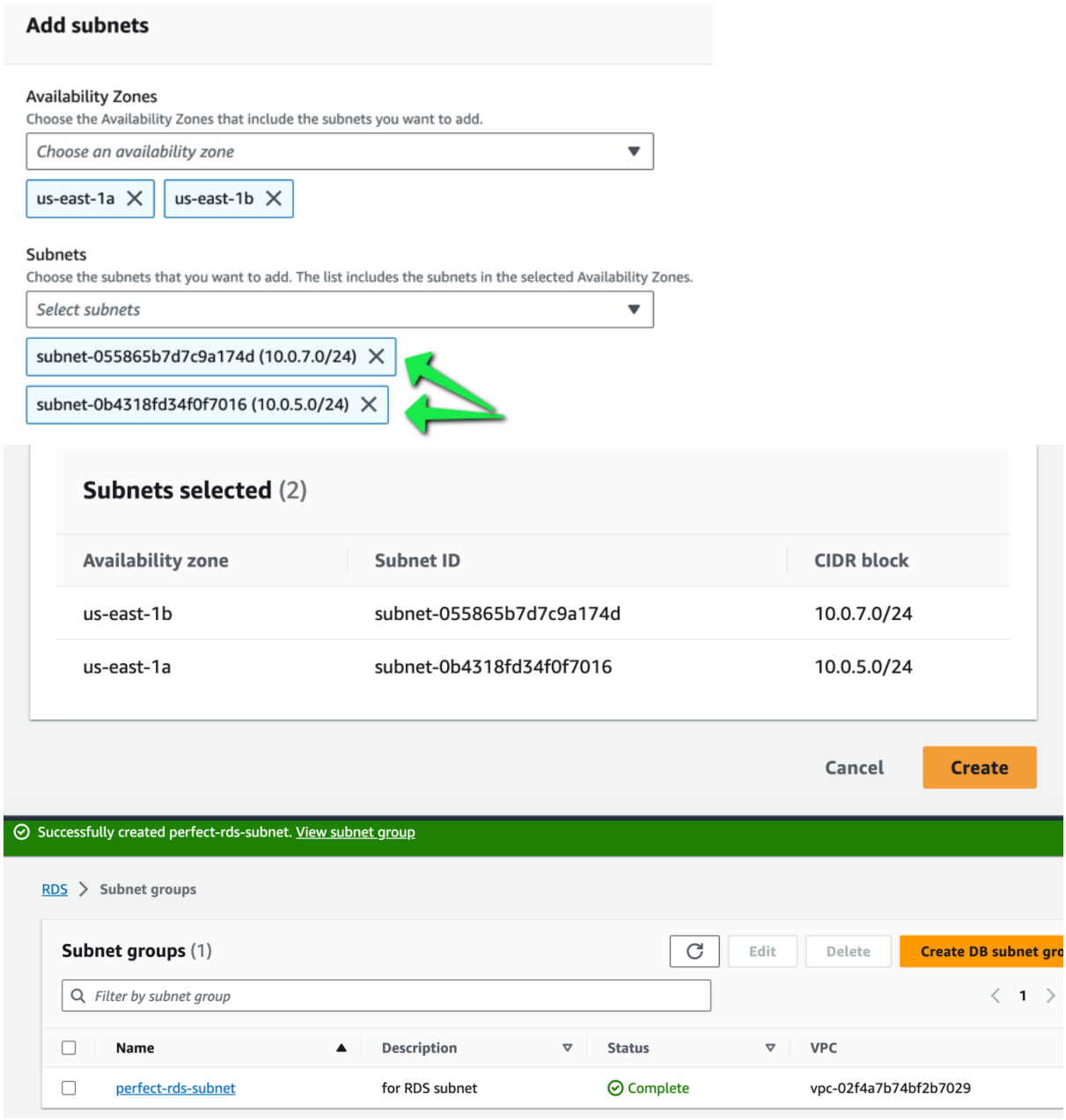




Please note: Ensure you only designate database access to only one who needs access to it .

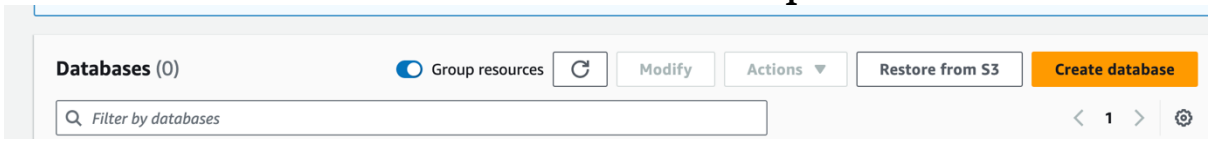
22) Next would be to create a DB subnet group for our private subnet



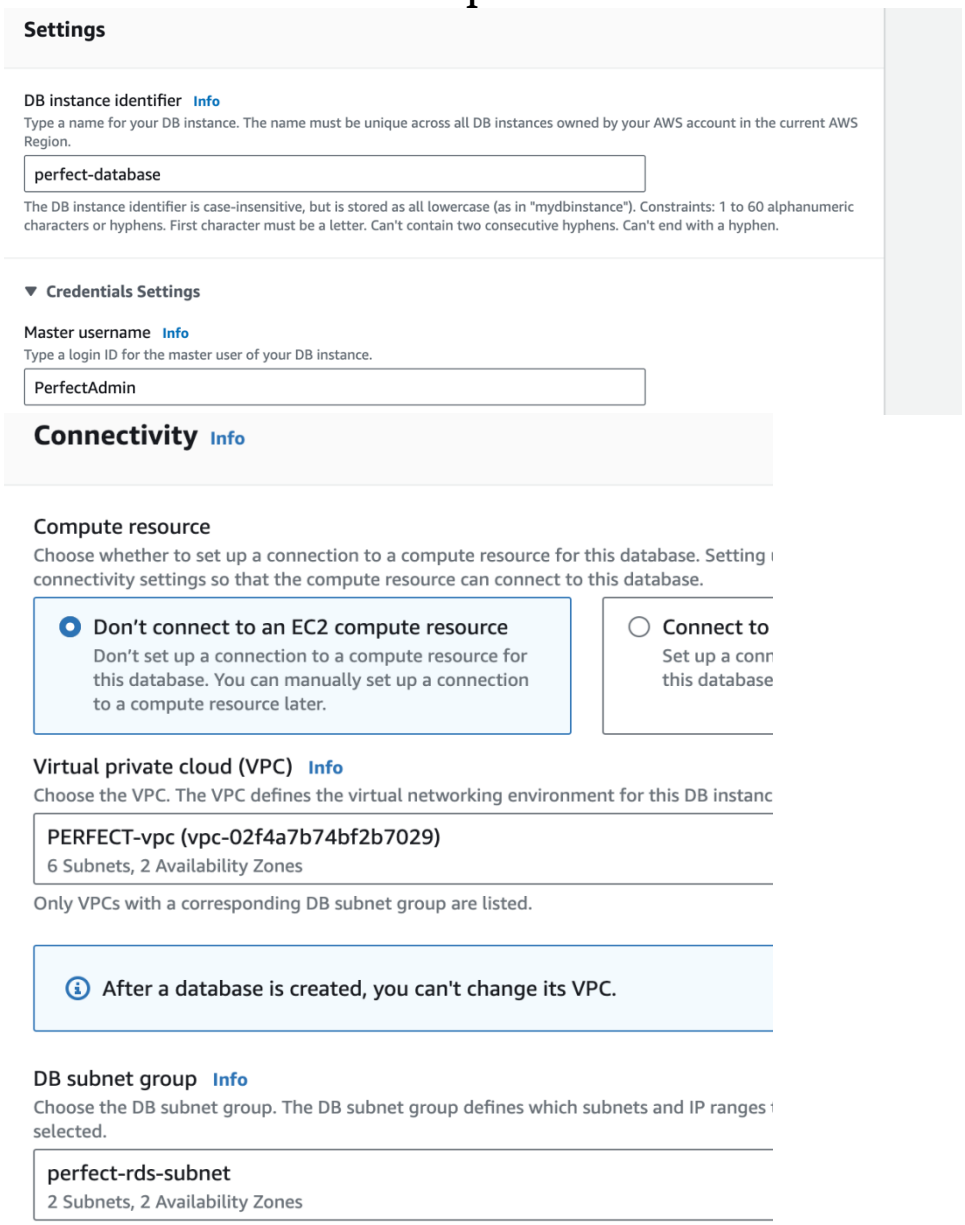


Subnet group successfully created .Next would be to create a database .

23) Create database .When creating it we would be choosing the free tier template as it won't give us access to be able to use the KMS key but if selecting the dev, test or production ,You would have access to the KMS and it comes with an expensive cost .



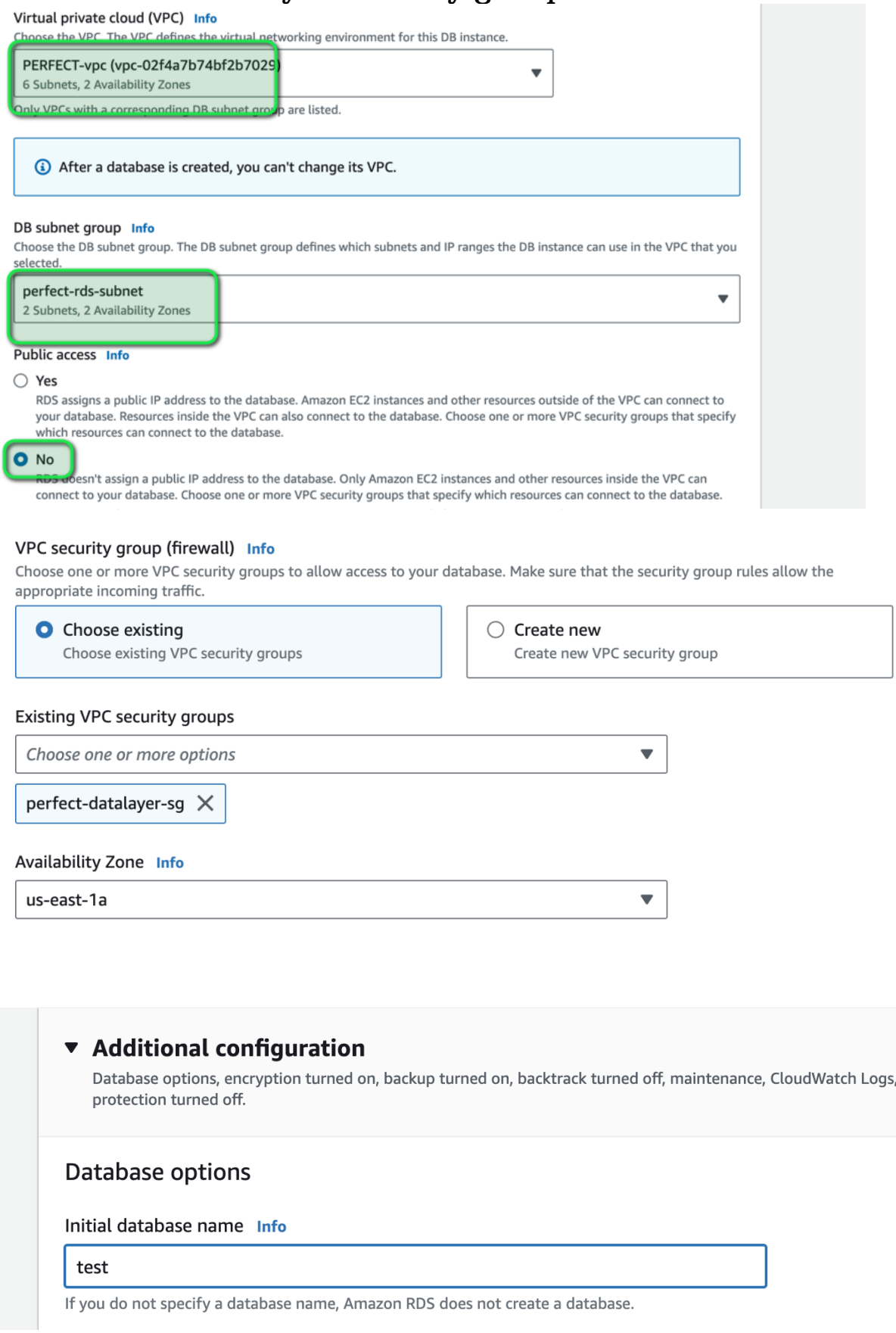
Choose username and password details .



Click No so that our RDS instance wouldn't be accessible .
The Amazon RDS Free Tier is available to you for 12 months. Each calendar month, the free tier will allow you to use the Amazon RDS resources listed below for free:

- 750 hrs of Amazon RDS in a Single-AZ db.t2.micro, db.t3.micro or db.t4g.micro Instance.

Choose the data layer security group



Database created.

24)Our next task would be to create an auto scaling group which would need some hosts of resources to be created namely.

a) AMI. b) Launch Template c)Target Group

d) Load balancer e) Autoscaling group

We would need to create a launch template but to do that we need to create 3 instances for our bastion, nginx and webservers and then we would be able to create 3 AMI respectively and we have to have some software installed inside them. Then we would come up with their user data. With this 2 ,then our launch template would be created

Name	Instance ID	Instance state	Instance type
Bastion	i-0e1a12f467cdb91a5	Running	t2.micro
Nginx	i-0876c40cfe5716499	Running	t2.micro
Webserver	i-0c4d3fe9f282b6b57	Running	t2.micro

25)For Bastion: The installation is below.

```
[ec2-user@Bastion ~]$ sudo su -
[root@Bastion ~]# yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm

yum install -y dnf-utils http://rpms.remirepo.net/enterprise/remi-release-8.rpm
Updating Subscription Management repositories.
Unable to read consumer identity

This system is not registered with an entitlement server. You can use subscription-manager to register.

Red Hat Enterprise Linux 9 for x86_64 - AppStream from RHUI (RPMs)          24 MB/s |  25 MB     00:01
Red Hat Enterprise Linux 9 for x86_64 - BaseOS from RHUI (RPMs)             24 MB/s |  14 MB     00:00
```

```
[root@Bastion ~]# yum install wget vim python3 telnet htop git mysql net-tools chrony -y
Updating Subscription Management repositories.
Unable to read consumer identity
```

WE installed python, ntp, net tools, vim, wget, telnet, epel-release-stop

```
[root@Bastion ~]# yum install wget vim python3 telnet htop git mysql net-tools chrony -y
Updating Subscription Management repositories.
Unable to read consumer identity

This system is not registered with an entitlement server. You can use subscription-manager to register.

Last metadata expiration check: 0:04:26 ago on Fri 13 Oct 2023 03:57:55 PM UTC.
Package python3-3.9.16-1.el9.x86_64 is already installed.
Package chrony-4.3-1.el9.x86_64 is already installed.
Dependencies resolved.
================================================================================
 Package              Architecture        Version                 Repository
================================================================================
Installing:
 git                  x86_64              2.39.3-1.el9_2          rhel-9-appstream-rhui-rpms
 htop                 x86_64              3.2.1-1.el8             epel
 mysql                x86_64              8.0.32-1.el9_2          rhel-9-appstream-rhui-rpms
 net-tools            x86_64              2.0-0.62.20160912git.el9 rhel-9-baseos-rhui-rpms
 telnet               x86_64              1:0.17-85.el9           rhel-9-appstream-rhui-rpms
 vim-enhanced         x86_64              2:8.2.2637-20.el9_1     rhel-9-appstream-rhui-rpms
```

```
Complete!
[root@Bastion ~]# systemctl start chronyd

systemctl enable chronyd
[root@Bastion ~]#
```

Once installed the chronyd is started we should create the AMI.

26)For Nginx: The installation is below.

We are carrying out the same installation but also adding the selinux configuration policies for the nginx server for the website to function properly.

We are also installing a self-signed certificate for our nginx and this is because the port must be set to 443 which is what nginx listens to.

```
[ec2-user@ip-172-31-21-30 ~]$ sudo hostname nginx
[ec2-user@ip-172-31-21-30 ~]$ bash
[ec2-user@nginx ~]$
```

```
[ec2-user@nginx ~]$ sudo su -
[root@nginx ~]# yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm

yum install -y dnf-utils http://rpms.remirepo.net/enterprise/remi-release-8.rpm
Updating Subscription Management repositories.
Unable to read consumer identity

This system is not registered with an entitlement server. You can use subscription-manager to register.

Red Hat Enterprise Linux 9 for x86_64 - AppStream from RHUI (RPMs)                                  43 MB/s |  25 MB     0
Red Hat Enterprise Linux 9 for x86_64 - BaseOS from RHUI (RPMs)                                     38 MB/s |  14 MB     0
```

```
[root@nginx ~]# yum install wget vim python3 telnet htop git mysql net-tools chrony -y

systemctl start chronyd

systemctl enable chronyd
Updating Subscription Management repositories.
```

We are carry out the same installation but also adding the selinux configuration policies for the webservers. this is because the port must be set to 443 which is what webserver listens to .

```
[root@nginx ~]# setsebool -P httpd_can_network_connect=1
setsebool -P httpd_can_network_connect_db=1
setsebool -P httpd_execmem=1
setsebool -P httpd_use_nfs 1
[root@nginx ~]#
```

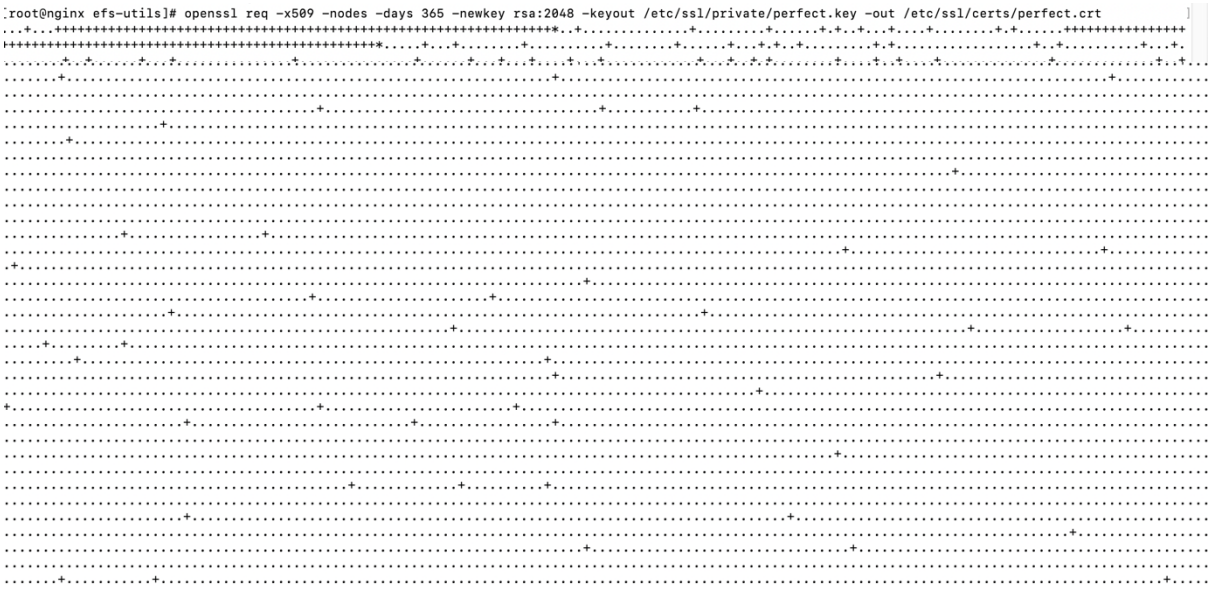
Please note that the reason we are performing it here is that our user-data exceed the 6kb limit .

```
[root@nginx ~]# git clone https://github.com/aws/efs-utils

cd efs-utils

yum install -y make

yum install -y rpm-build

make rpm

yum install -y  ./build/amazon-efs-utils*rpm
Cloning into 'efs-utils'...
remote: Enumerating objects: 1449, done.
remote: Counting objects: 100% (52/52), done.
remote: Compressing objects: 100% (23/23), done.
remote: Total 1449 (delta 38), reused 29 (delta 29), pack-reused 1397
```

Installing self-signed certificate for nginx

```
Country Name (2 letter code) [XX]:UK
State or Province Name (full name) []:Manchester
Locality Name (eg, city) [Default City]:Manchester
Organization Name (eg, company) [Default Company Ltd]:Devops
Organizational Unit Name (eg, section) []:Devops
Common Name (eg, your name or your server's hostname) []:ip-172-31-21-30.ec2.internal
Email Address []:mrincredible752@outlook.com
[root@nginx efs-utils]#
```

Creating a dhparam and wait for a couple of minute to get it fully created .



Please confirm the .key and .crt files are present.
Nginx successfully done and we would create AMI next .

27)For Webserver : The installation is below
We are carrying out the same installation but also adding the selinux configuration policies for the webserver for the website to function properly.

We are also installing a self-signed certificate for our Apache webserver and this is because the port must be set to 443 which is what webserver listens to.

```
[ec2-user@ip-172-31-28-13 ~]$ sudo hostname webservers
[ec2-user@ip-172-31-28-13 ~]$ bash
 [ec2-user@webservers ~]$
```

After installation we would edit the Apache configuration file

```
[root@webservers ~]# vi /etc/httpd/conf.d/ssl.conf
[root@webservers ~]#
```

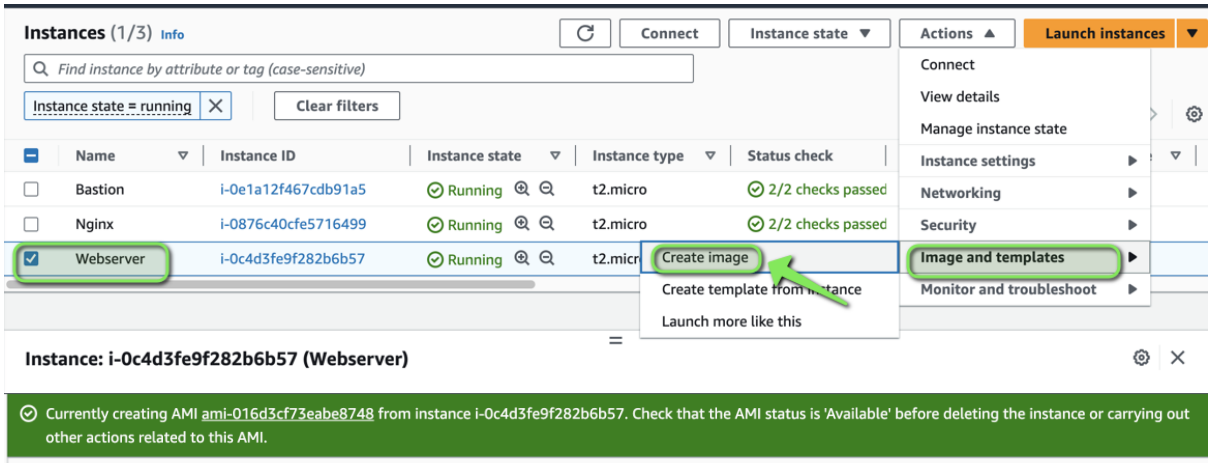
Change from localhost.crt to perfect.cert

Change from localhost.key to perfect.key

```
#   require an ECC certificate which can also be configured in
#   parallel.
SSLCertificateFile /etc/pki/tls/certs/perfect.crt

#   Server Private Key:
#   If the key is not combined with the certificate, use this
#   directive to point at the key file.  Keep in mind that if
#   you've both a RSA and a DSA private key you can configure
#   both in parallel (to also allow the use of DSA ciphers, etc.)
#   ECC keys, when in use, can also be configured in parallel
SSLCertificateKeyFile /etc/pki/tls/private/perfect.key

#   Server Certificate Chain:
#   Point SSLCertificateChainFile at a file containing the
```

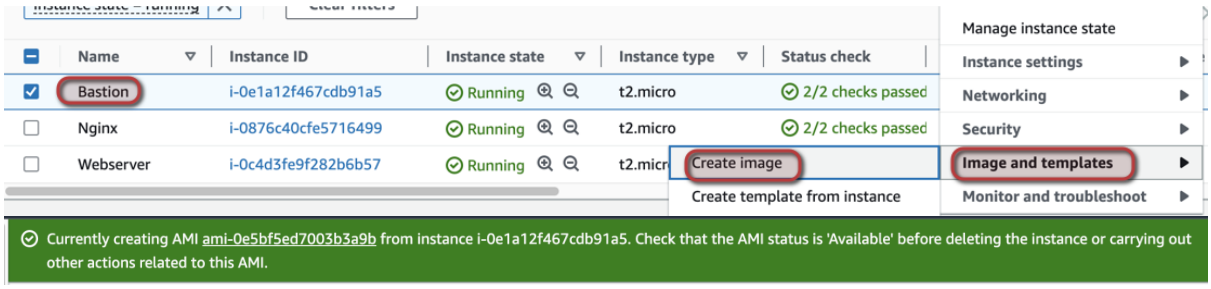
Once saved we would be creating an AMI from it .
Hence we are creating AMI for each of the 3 server .

28)Webserver AMI



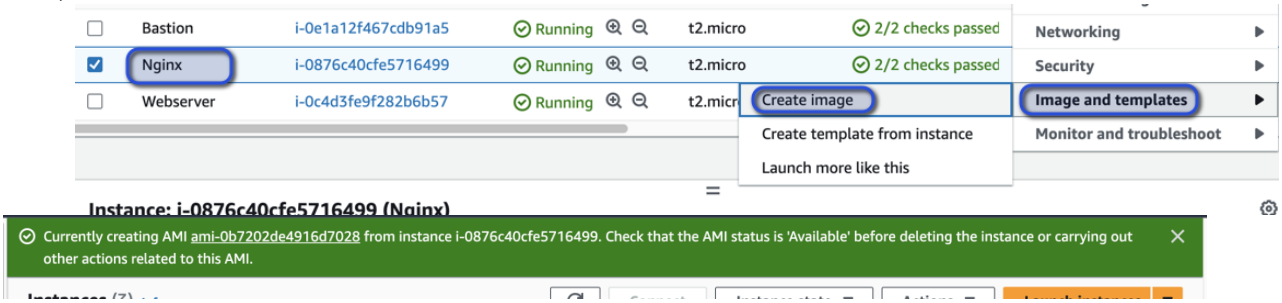
Webserver AMI successfully created

30) Bastion AMI successfully created



Bastion AMI successfully created

31)NGINX AMI

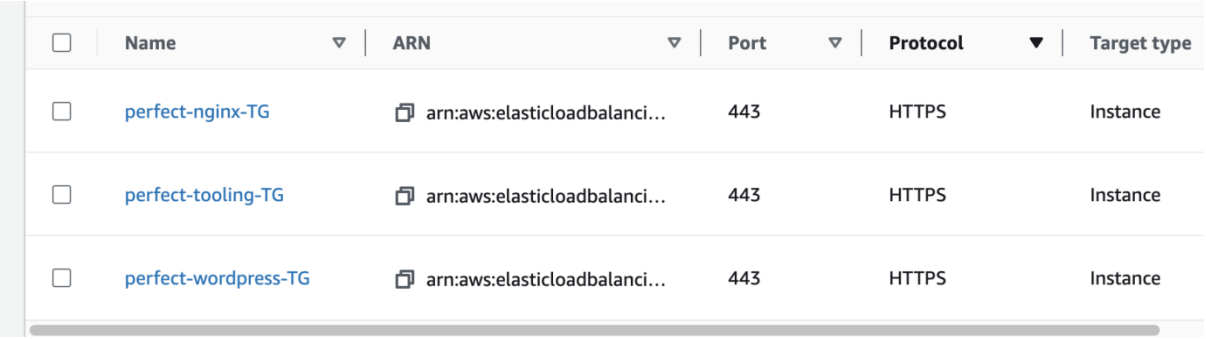


NGINX AMI successfully created

32) Create Target group for the Nginx, WordPress Server and WordPress Tooling Webserver.

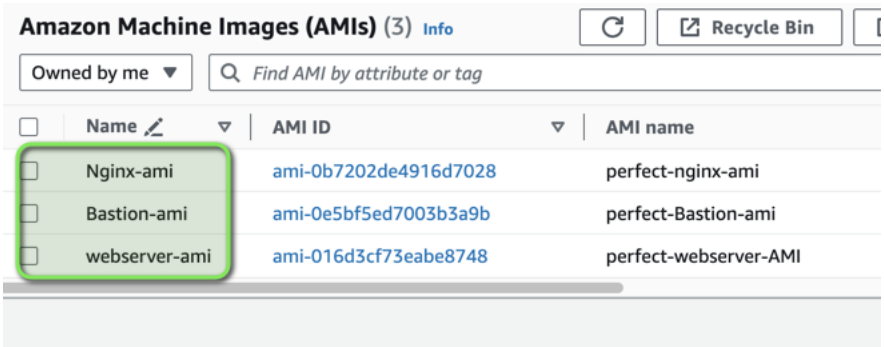
Please note we are not creating any target group for Bastion server because it is not behind any of the load balancers

All target groups created

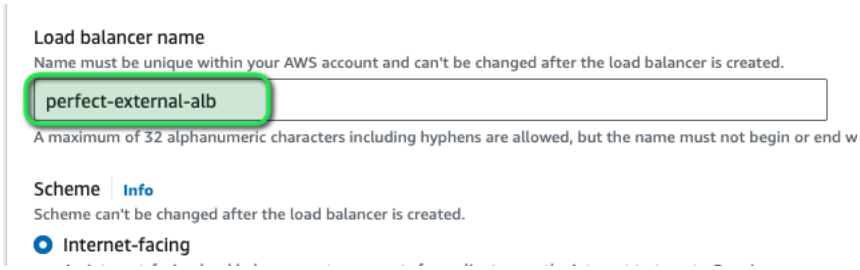


Name	ARN	Port	Protocol	Target type
perfect-nginx-TG	arn:aws:elasticloadbalanci...	443	HTTPS	Instance
perfect-tooling-TG	arn:aws:elasticloadbalanci...	443	HTTPS	Instance
perfect-wordpress-TG	arn:aws:elasticloadbalanci...	443	HTTPS	Instance

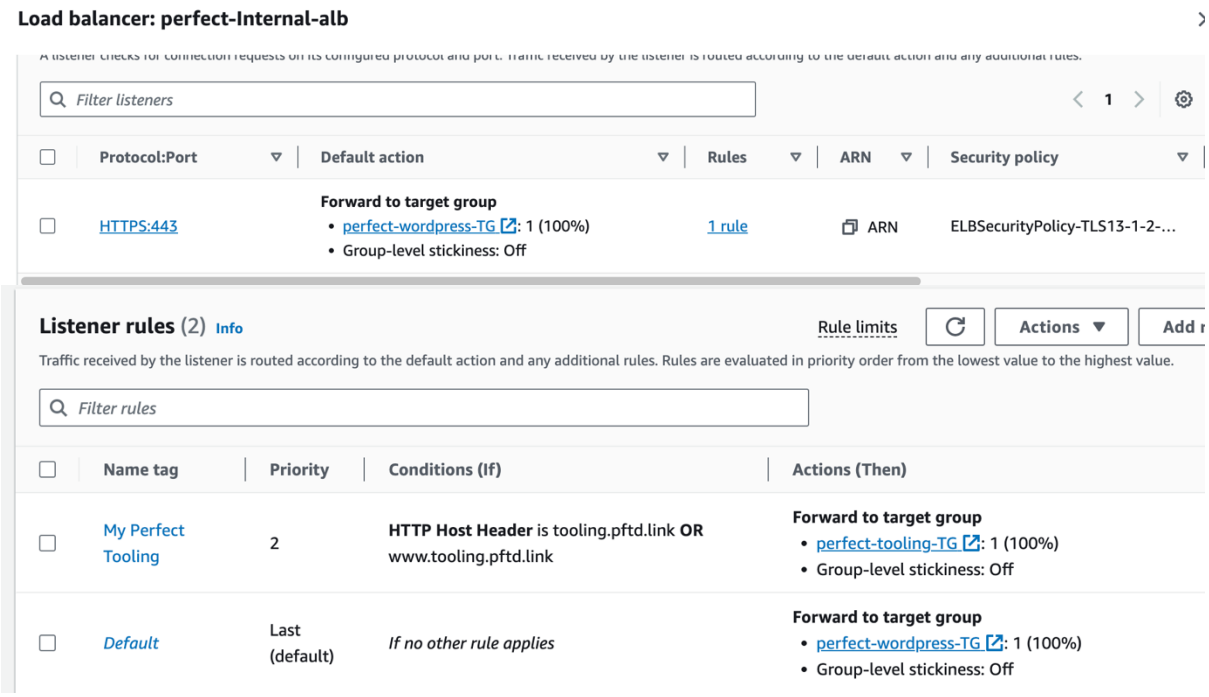
All AMI created



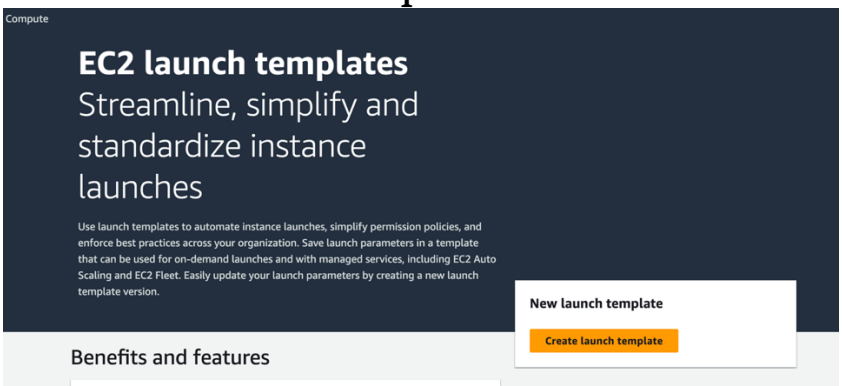
33)Create the external and internal load balancers and ensure they are on https :443



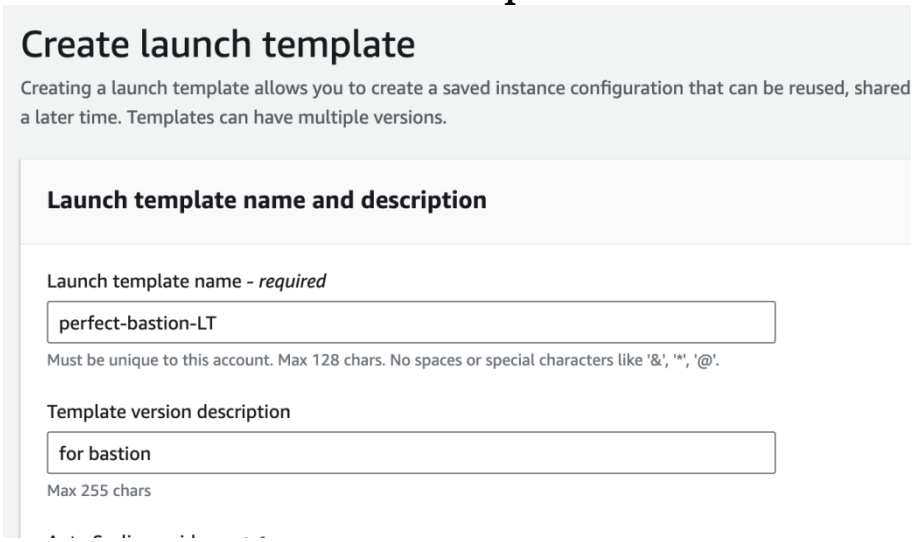
34)Create listener rules for the internal load balancer for WordPress and tolling



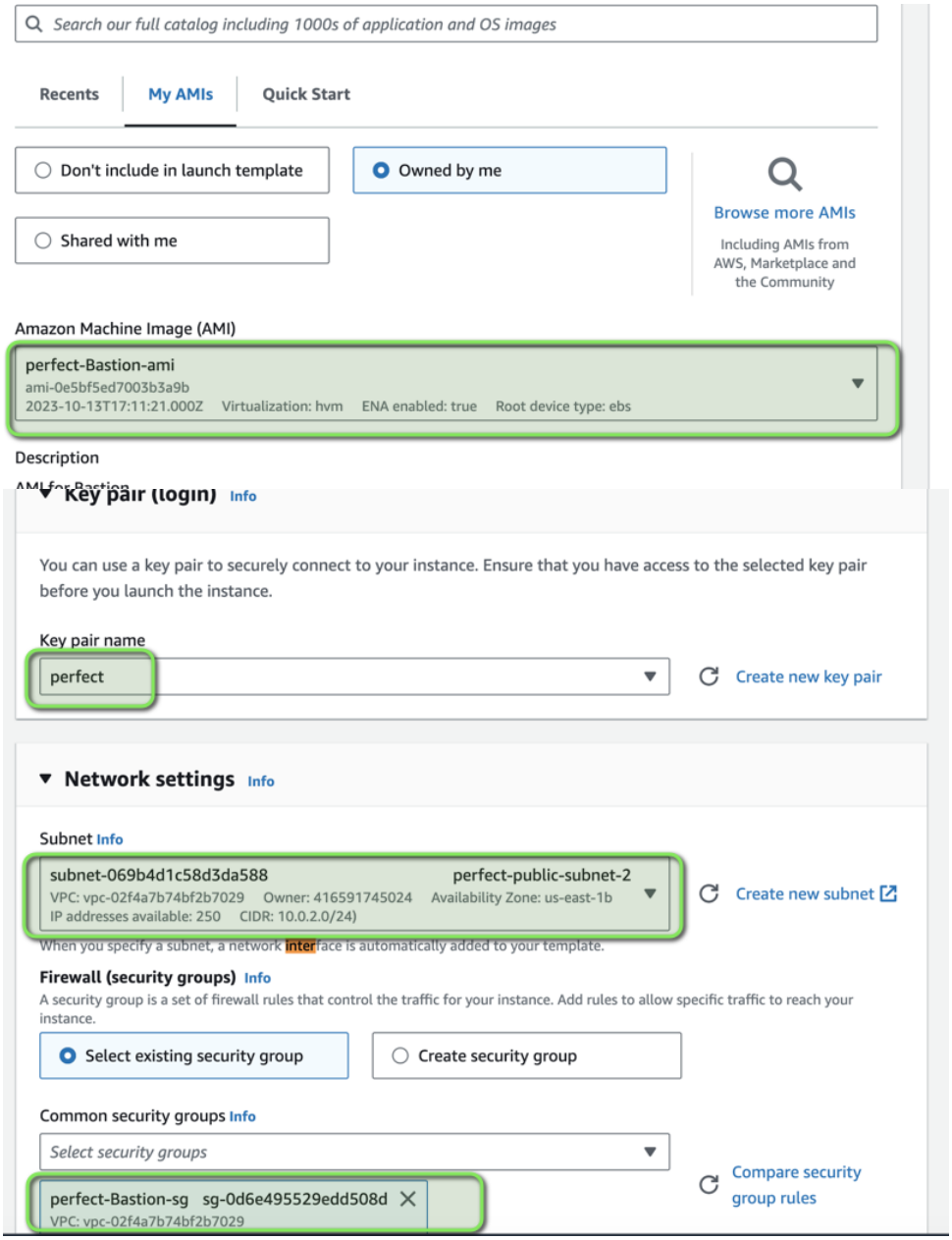
35)Create Launch Templates



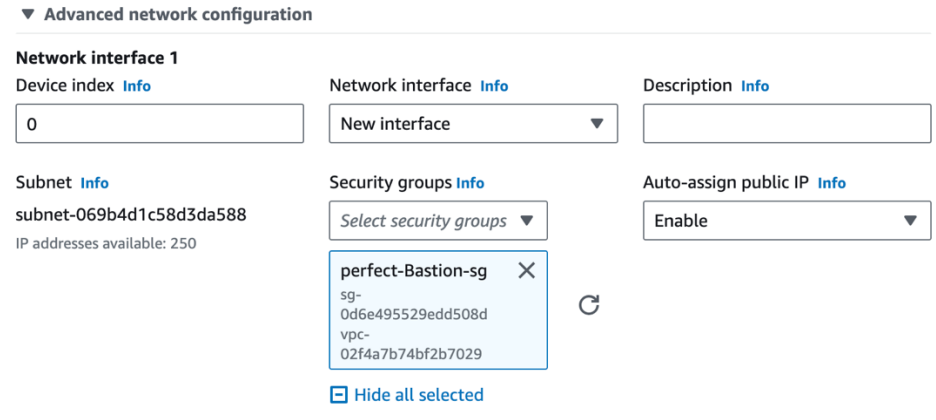
a) For Bastion Launch Template



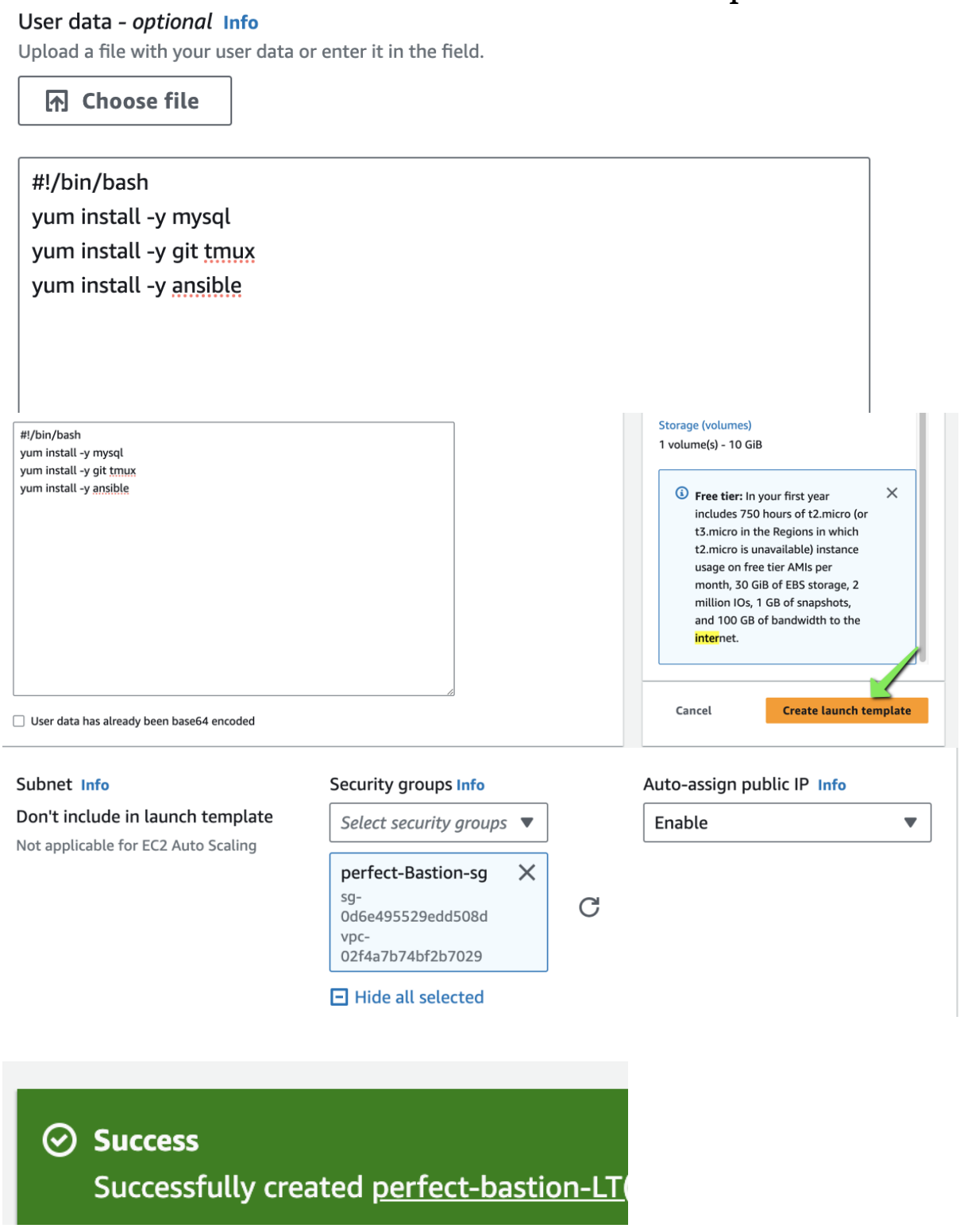
Select your Bastion AMI, Instance type ,Key Pair and in a public subnet



Add a network interface ,Add a security group and enable public IP

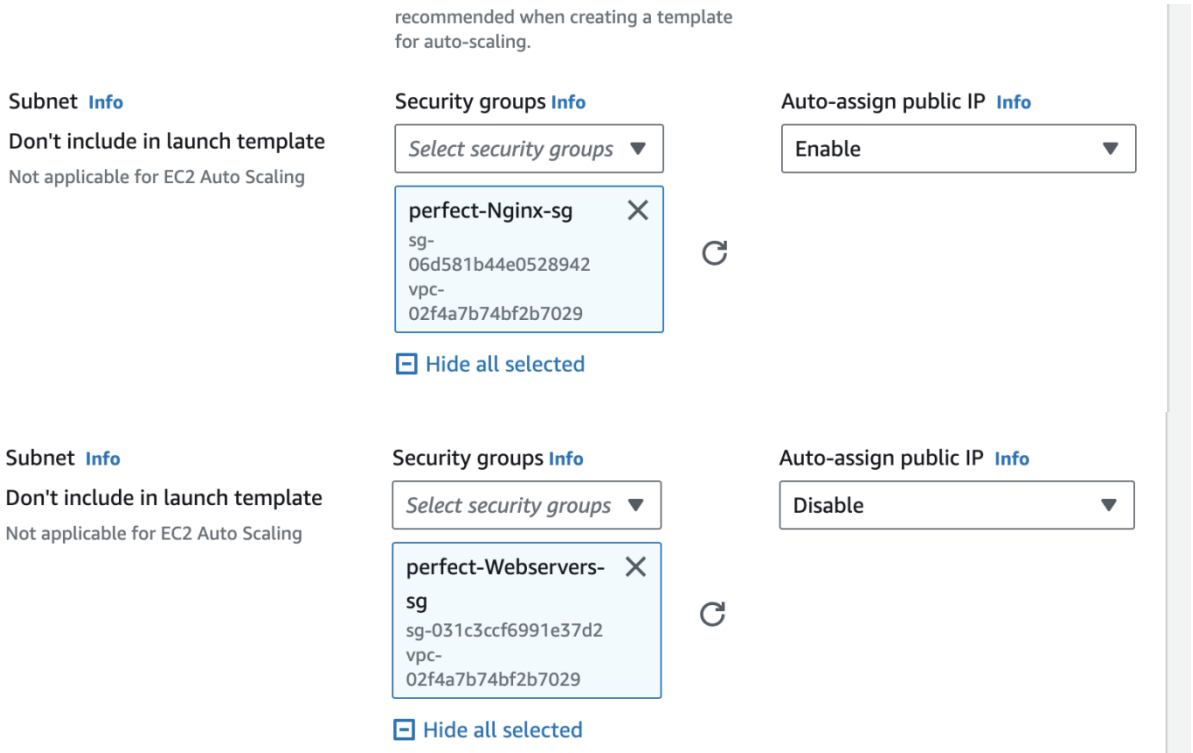


Add the user data and create the Launch template.



Bastion Launch Template created.

b) For Nginx Launch Template, perform the same task but ensure the auto assign IP is disabled.



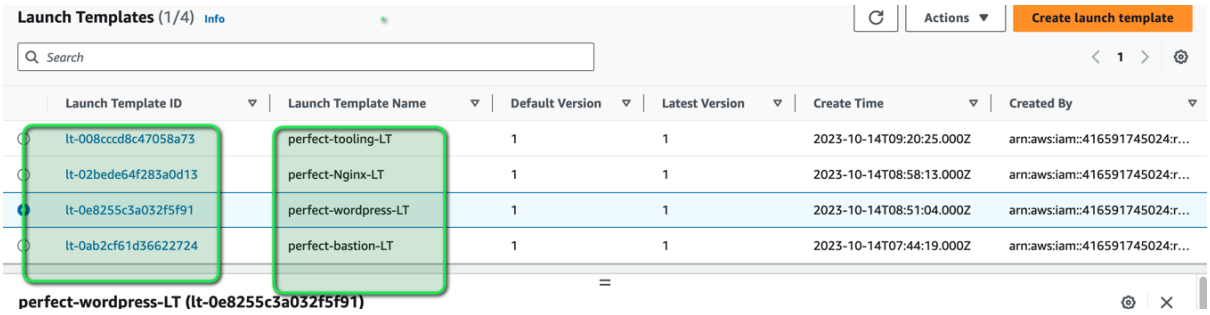
Please note : For WordPress :It would require mounting on EFS and this can be gotten when you navigate to the EFS access point by clicking “attach” button

sudo mount -t efs -o tls, accesspoint=fsap-072cbbfc6545251a0 fs-076e78cbfaa0bc3be:/ /var/www/

Also navigate to the database and copy the endpoint to add into the user data .After making this adjustment then it can be created .

```
#!/bin/bash
mkdir /var/www/
sudo mount -t efs -o tls,accesspoint=fsap-072cbbfc6545251a0 fs-076e78cbfaa0bc3be:/ /var/www/
yum install -y httpd
systemctl start httpd
systemctl enable httpd
yum module reset php -y
yum module enable php:remi-7.4 -y
yum install -y php php-common php-mbstring php-opcache php-intl php-xml php-gd php-curl php-mysqlnd php-fpm php-json
systemctl start php-fpm
systemctl enable php-fpm
wget http://wordpress.org/latest.tar.gz
tar xzvf latest.tar.gz
rm -rf latest.tar.gz
cp wordpress/wp-config-sample.php wordpress/wp-config.php
mkdir /var/www/html/
cp -R /wordpress/* /var/www/html/
cd /var/www/html/
touch healthstatus
sed -i "s/localhost/perfect-database.[redacted].amazonaws.com/g" wp-config.php
sed -i "s/username_here/PerfectAdmin/g" wp-config.php
sed -i "s/password_here/admin12345/g" wp-config.php
sed -i "s/database_name_here/wordpressdb/g" wp-config.php
chcon -t httpd_sys_rw_content_t /var/www/html/ -R
systemctl restart httpd
```

The 4 launch templates are created using their respective AMI and user data.

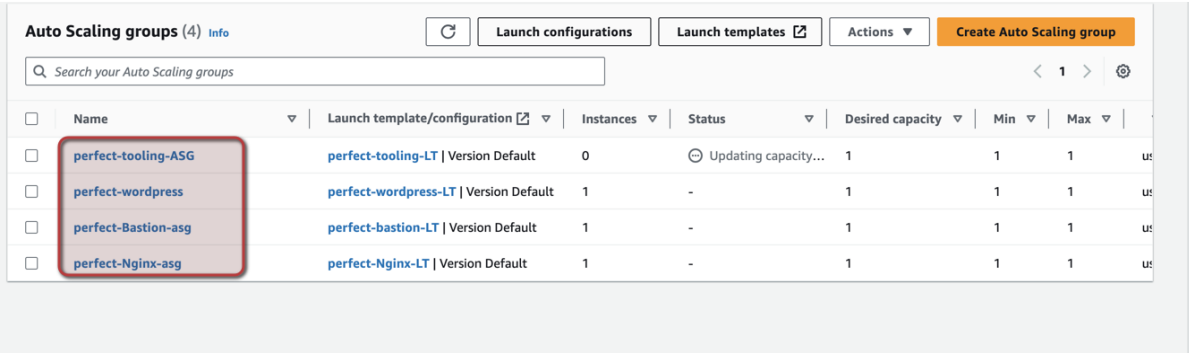


36) Create Auto scaling group. For bastion and nginx first and then proceed to create the tooling and webserver. But we need to create the actual data base first and then create it .

37)We would be using the SSH-AGENT to get into the instance as shown below to have access and create the tooling data base and the WordPress database.

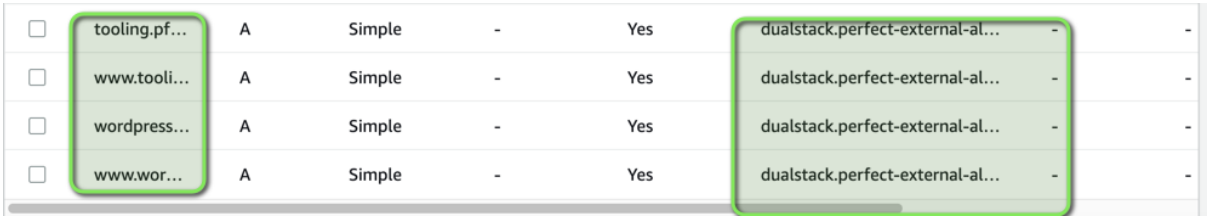
```
[ec2-user@ip-10-0-0-190 ~]$ mysql -h perfect-database.cnfvqr8ijadp.us-east-1.rds.amazonaws.com -u PerfectAdmin -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 18
Server version: 8.0.33 Source distribution

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> create database wordpressdb;
Query OK, 1 row affected (0.01 sec)

mysql> create database toolingdb;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| test               |
| toolingdb          |
| wordpressdb        |
+--------------------+
7 rows in set (0.01 sec)
```

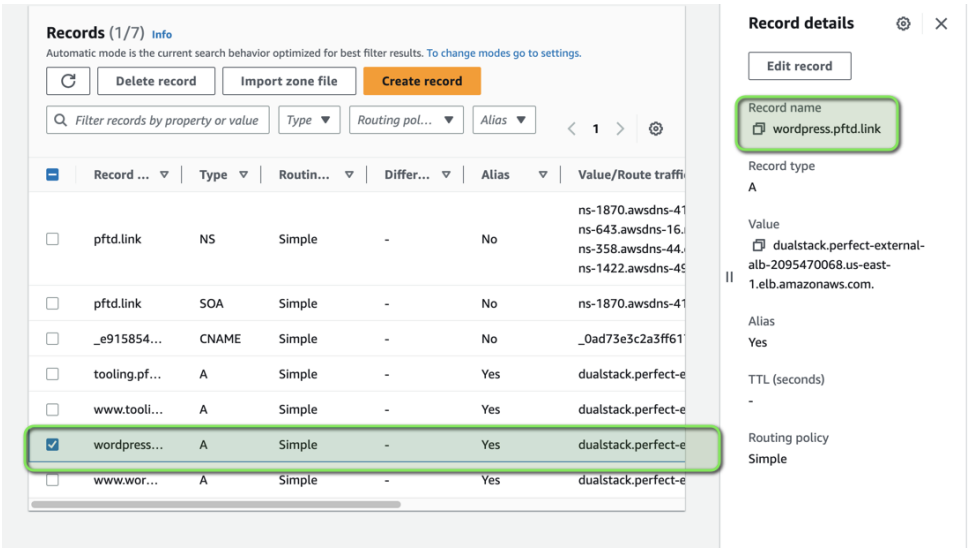
After creating we now have the 4 Auto scaling groups



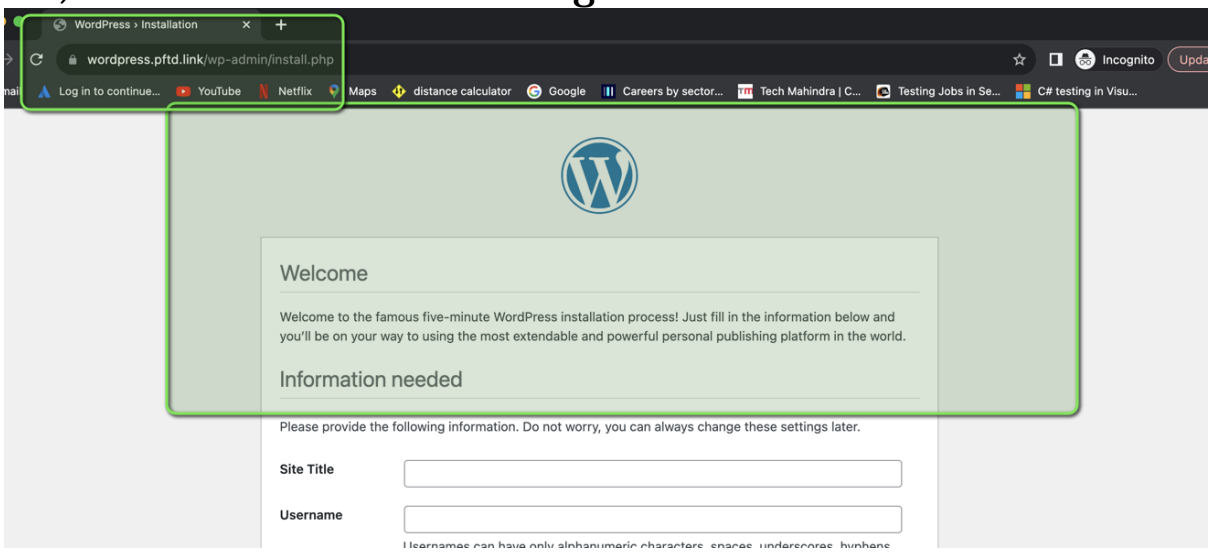
38)Create records for our tooling webserver and WordPress webserver as shown below. All traffic from the website should be forwarded to the external load balancer.

Add www.tooling.pftd.link or tooling.pftd.link
Add www.wordpress.pftd.link or wordpress.pftd.link as shown below.



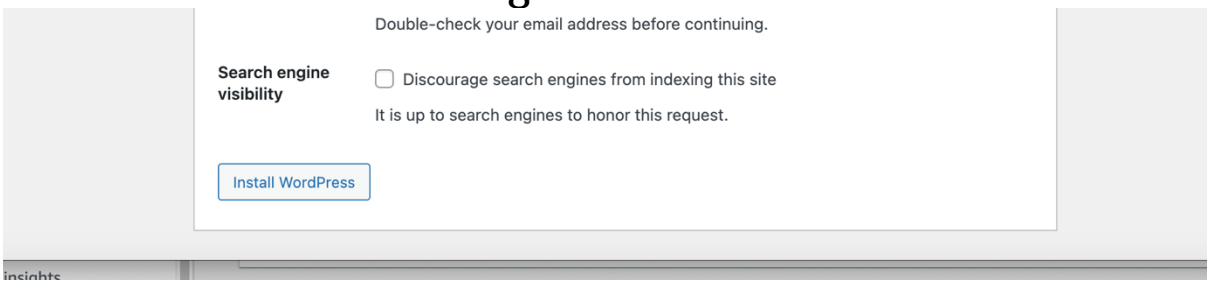


39)Launch the website using the link for WordPress.

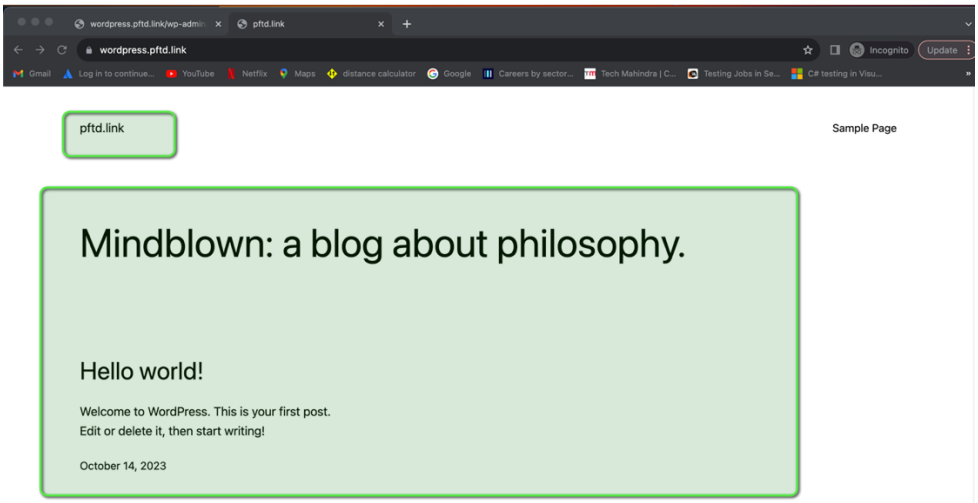


WordPress Launched successfully.

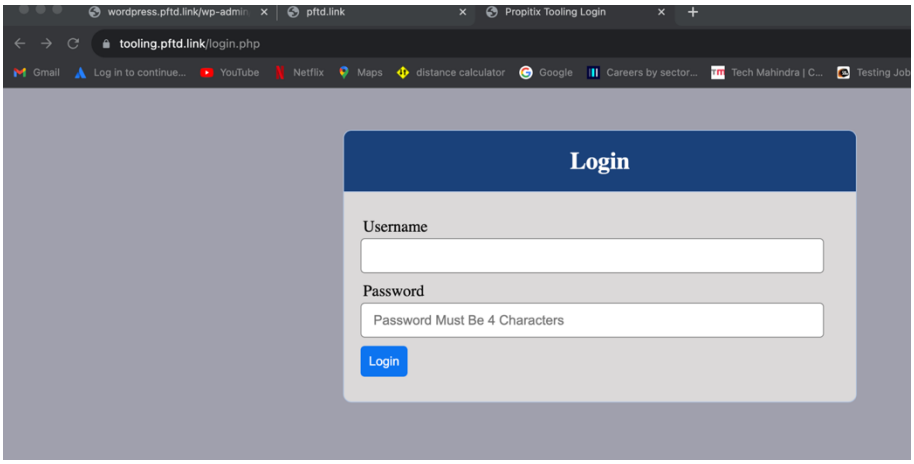
40)Install WordPress and log in the credentials.



Our WordPress is successfully launched.



41)Launch the website using the link for tooling.



42)Tooling website launched successfully.

In Conclusion, we have successfully launched a website. This process starts when the client (user or DevOps Engineer) types the website on the browser which goes through the route53 DNS to the internet gateway into the External load balancer which sends the traffic to the Nginx reverse Proxy then routes into the Internal Load Balancer .
The internal Load balancer checks the host headers and if it recognizes it is default, it routes to the WORDPRESS but if otherwise it routes to the tooling website and then target renders the website for our view .

Congratulations We have successfully launched 2 company website using AWS Cloud Solution.

Please remember to delete all resources once the project is completed.

Thanks