Amazon Virtual Private Cloud (Amazon VPC)

Amazon Virtual Private Cloud (Amazon VPC) enables us to launch AWS resources into a virtual network that we've defined.

➤ Amazon VPC is the networking layer for Amazon EC2 Instances.

➤ For Fresh account Amazon have default VPC Network.

➤ Best Practice to always create our own VPC with our own Settings and Configuration.

Few Key Points and Terminology:

➤ Virtual private cloud (VPC): A virtual network dedicated to our AWS account.

➤ Subnet: A range of IP addresses in our VPC.

➤ Route table: A set of rules, called routes, that are used to determine where network traffic is directed.

➤ Internet gateway: A gateway that we attach to our VPC to enable communication between resources in our VPC and the internet.

➤ VPC endpoint: Enables us to privately connect our VPC to supported AWS services and VPC endpoint services powered by Private Link without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection.

What user can do with VPC:

➤ Launch instances in a subnet of our choice. we can choose our own subnet addressing.

➤ we can assign custom IP address ranges in each subnet.

➤ we can configure route tables between subnets.

➤ we can create an internet gateway and attach it to our VPC.

➤ It provides much better security control over our AWS resources.

➤ we can assign security groups to individual instances.

Amazon Virtual Private Cloud (Amazon VPC)

➤ we also have subnet network access control lists (ACLS).

➤ For Small or Medium Setup One VPC will be enough.

➤ An Instance Launched in one VPC can never communicate to Instance Launched in another VPC via Private IP.

➤ Public IP is must to setup the inter VPC communication But Two VPCs can be linked via PVC Peering.

VPC Peering:

➤ VPC Peering is a networking connection that allows us to connect one VPC with another VPC through a direct network route using private IP addresses. ➤ Instances behave as if they were on the same private network.

➤ we can peer VPC's with other AWS accounts as well as other VPCs in the same account.

➤ Peering is in a star configuration, i.e., 1 VPC peers other 4 VPCs.

we can peer between regions. Suppose we have one VPC in one region and other VPC in another region, then we can peer the VPCs between different regions. One VPC can be connected via 1 or more VPCs.

Subnet in VPC:

Virtual private cloud (VPC) is a virtual network dedicated to our AWS account. It is logically isolated from other virtual networks in the AWS Cloud.

➤ When we create a VPC, we must specify a range of IPv4 addresses for the VPC in the form of a Classless InterDomain Routing (CIDR) block 10.0.0.0/16 is the primary CIDR block for your VPC.

➤ VPC spans all of the Availability Zones in the Region. After creating a VPC, we can add one or more subnets in each Availability Zone.

➤ If a subnet's traffic is routed to an internet gateway, the subnet is known as a public subnet.

➤ If a subnet doesn't have a route to the internet gateway, the subnet is known as a private subnet.

➤ VPC and subnet sizing for IPv4: 10.0.0.0 - 10.255.255.255 (10/8 prefix) - User VPC must be /16 or smaller,

for example, 10.0.0.0/16.

172.16.0.0 - 172.31.255.255 (172.16/12 prefix)-User VPC must be /16 or smaller, for example, 172.31.0.0/16.

➤ 192.168.0.0 - 192.168.255.255 (192.168/16 prefix)-User VPC can be smaller, for example 192.168.0.0/20.

To add a CIDR block to our VPC, the following rules apply:

➤ The allowed block size is between a /28 netmask and /16 netmask.

➤ CIDR block must not overlap with any existing CIDR block that's associated with the VPC. User cannot increase or decrease the size of an existing CIDR block.

Security Group in AWS

➤ Security group acts as a virtual firewall for your instance to control inbound and outbound traffic.

➤ Upto 5 SGs can be assigned to Instance in AWS.

➤ SGs are Instance Level not Subnet Level.

Basics of Security Group in AWS

➤ User can specify allow rules, but not deny rules.

➤ User can specify separate rules for inbound and outbound traffic.

➤ Security group rules enable us to filter traffic based on protocols and port numbers.

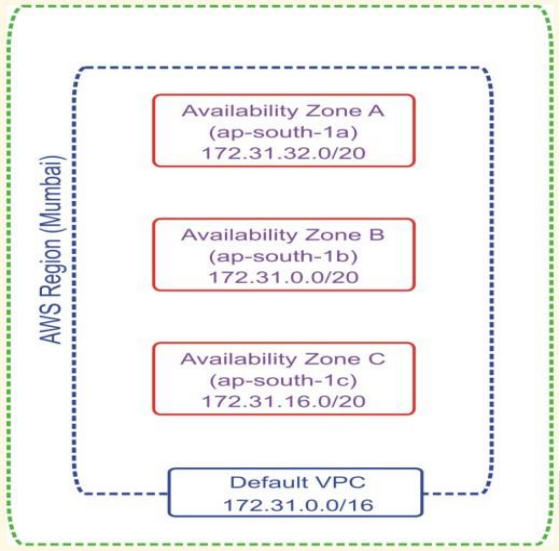
➤ By default, a Security Group don't have any Inbound Rule.

➤ By default, a security group includes an outbound rule that allows all outbound traffic.

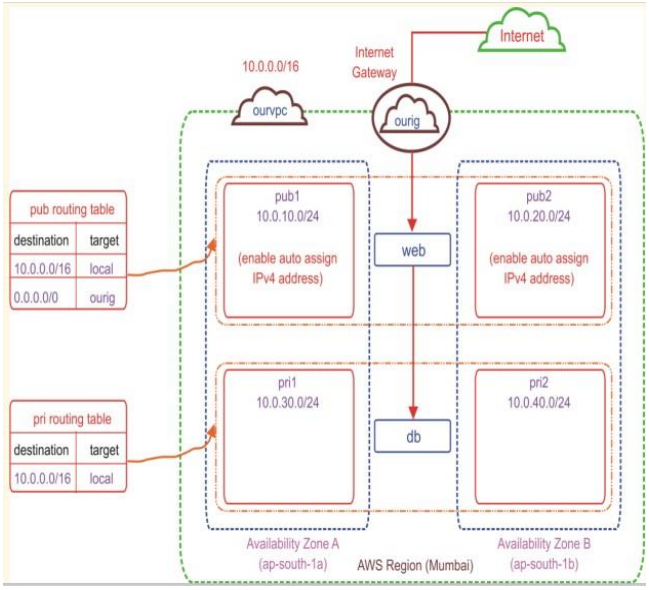
There are quotas on the number of security groups that we can create per VPC, the number of rules that we can add to each security group, and the number of security groups that we can associate with a network interface.

Lab Guide:

Lab-1: Understanding the Default VPC



Lab-2: Creating the Custom VPC with private and public subnets



Amazon Virtual Private Cloud (Amazon VPC)

1. Create VPC with 10.0.0.0/16 CIDR

2. Create Public subnet-1 within the custom VPc with CIDR 10.0.10.0/24 in the region ap-south-1a

3. Create public subnet-2 within the custom VPc with CIDR 10.0.20.0/24 in the region ap-south-1b

4. Create private subnet-1 within the custom VPC with CIDR 10.0.30.0/24 in the region ap-south-1a

5. Create private subnet-3 within the custom VPc with CIDR 10.0.40.0/24 in the region ap-south-1b

6. Create Internetgateway with in the custom VPC

7. Create Route Tables for public subnets and private subnets and attach it to VPC

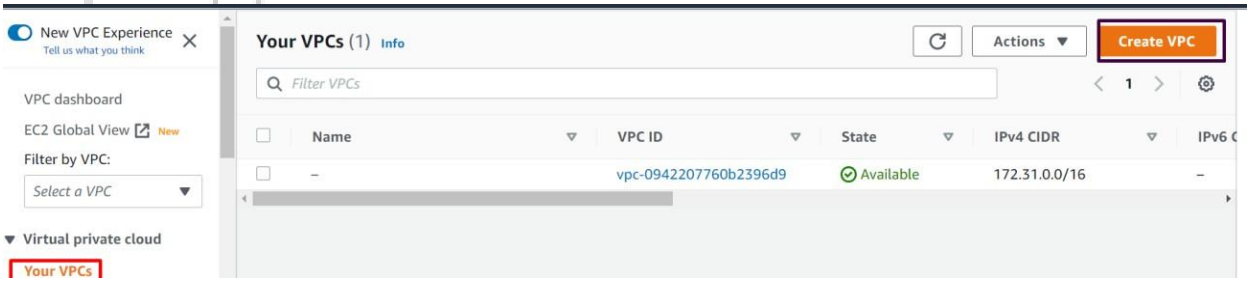
8. Edit the route table of the public subnet and attach the Internetgateway.

9. Create an EC2 machine in public subnet and install nginx in it, we whould be able to connect to the machine and access the nginx application.

10. Create an ec2 instance in private subnet we should not be able to connect to it.

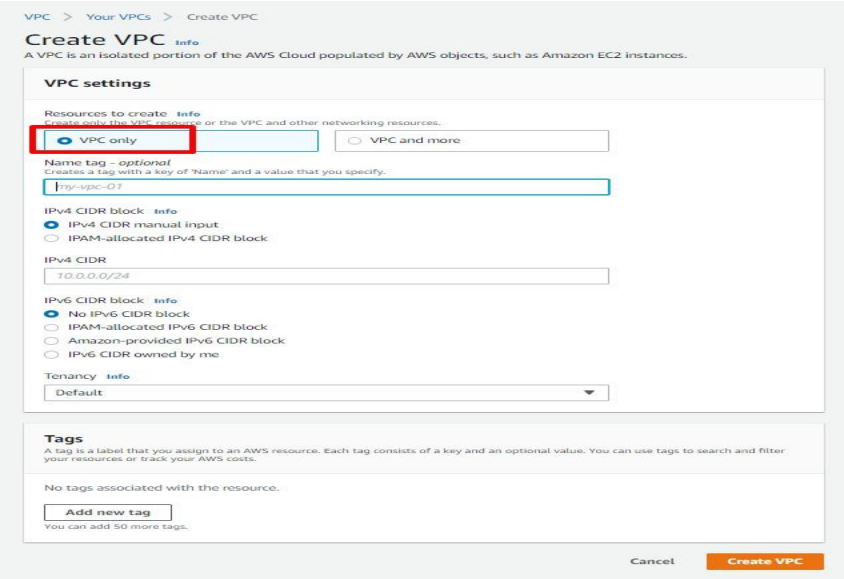
Creating the VPC

Step-1:Click on Create VPC button in the VPC dashboard as shown below:

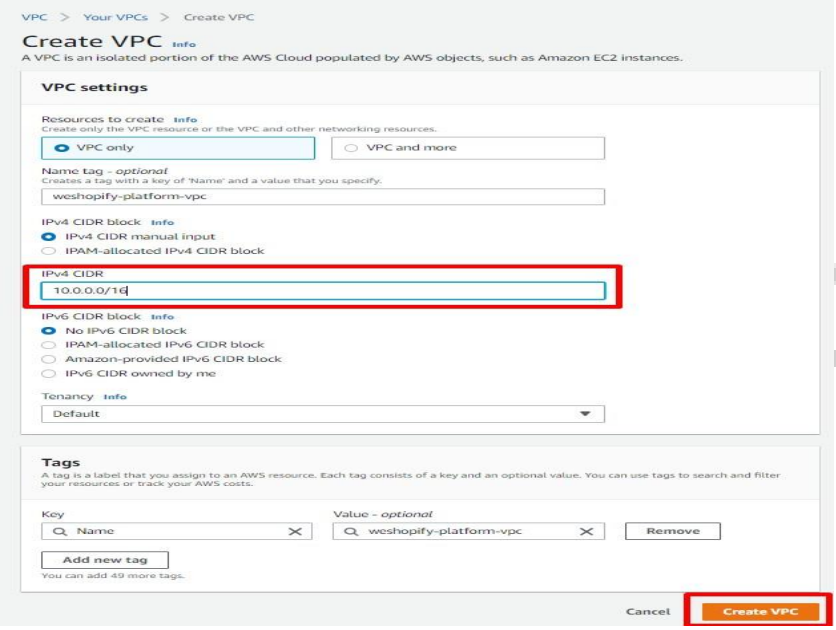


Step-2: when we click on Create VPC button as shown above, we will get the below screen to create the VPC.

Amazon Virtual Private Cloud (Amazon VPC)



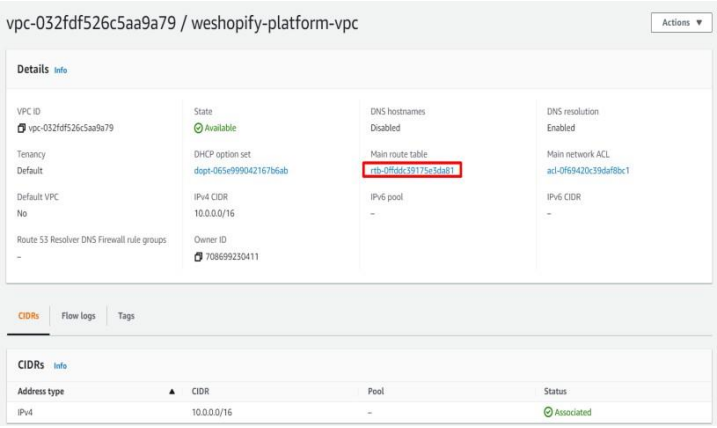
Step-3: enter the details as below to create the VPC



Here IPV4CIDR choose as 10.0.0.0/16 to generate the 32-16=16(2^16=65536 ip addresses)

Step-4: As soon as we click on Create VPC button above the VPC will be created with the default route table as main as shown below:

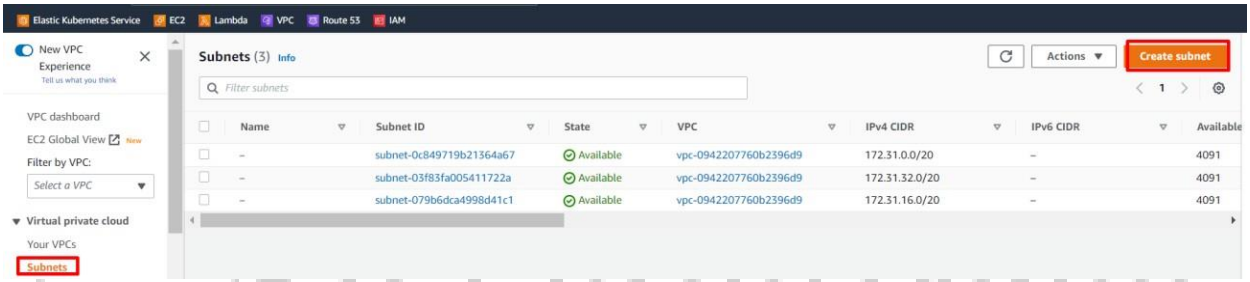
Amazon Virtual Private Cloud (Amazon VPC)



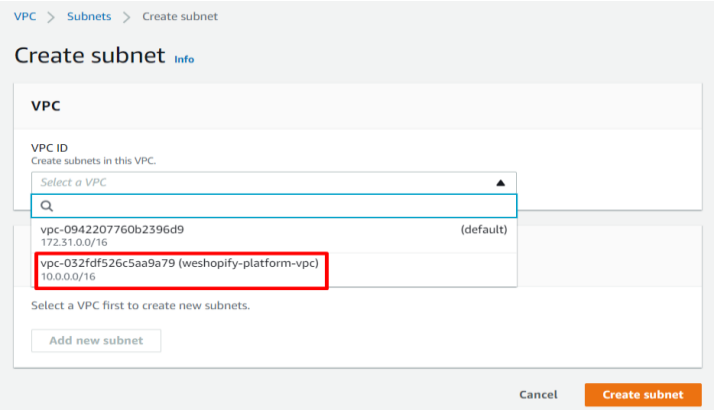
Note: here we must have to change the default route table to the one we gonna create in next further steps.

Creating the Subnets

Step-1:Click on Subnets and click on create subnet button as shown below:

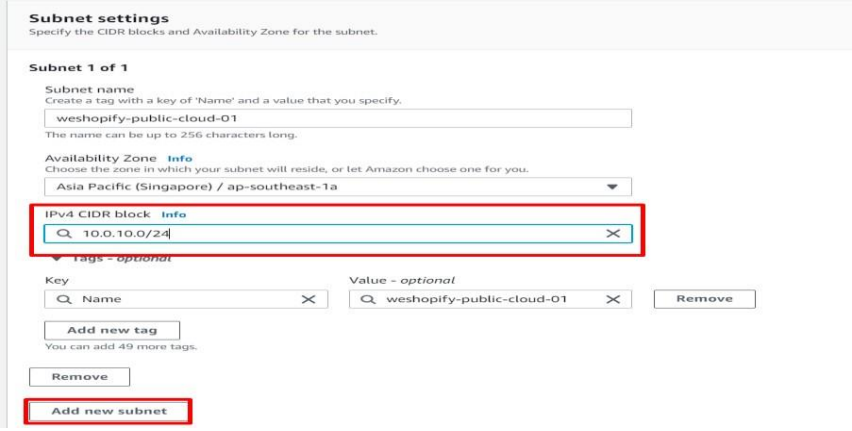


Step-2: As soon as we click on Create Subnet button we will see the below screen to create the customized subnets.

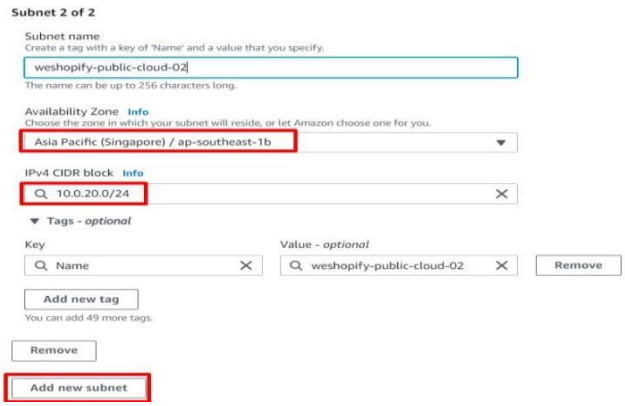


Step-3: Choose the Custom VPC I.e. weshopify-platform-vpc from the drop down as shown above and the following screen will be opened for us to create the subnets.

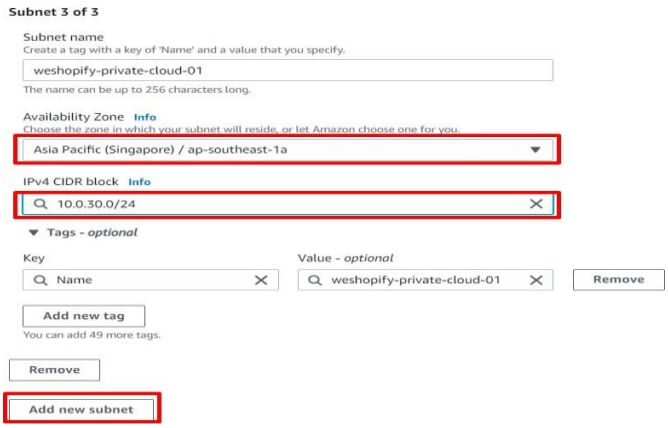
1. Creating the public subnet-1



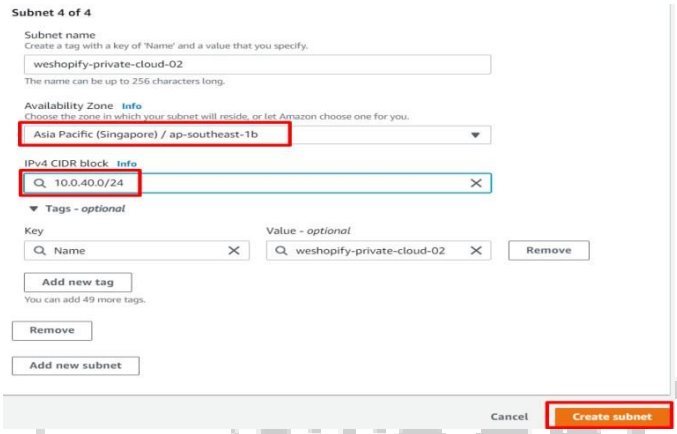
2. Creating the public subnet-2



3. Creating the private subnet-01



4. <u>Creating the private subnet-02</u>



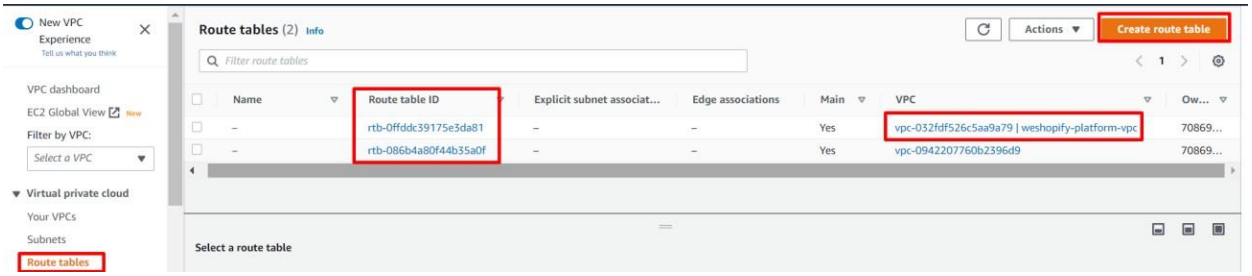
As soon as we click on create subnet button as shown above, we will be able to see the list of subnets created as shown below:

Name	Subnet ID	State	VPC	IPv4 CIDR
weshopify-private-cloud-01	subnet-0d59a51d44c960fd3	Available	vpc-032fdf526c5aa9a79 | weshopify-platform-vpc	10.0.30.0/24
weshopify-private-cloud-02	subnet-07b9cd718da393690	Available	vpc-032fdf526c5aa9a79 | weshopify-platform-vpc	10.0.40.0/24
weshopify-public-cloud-01	subnet-06f119134bc2f9879	Available	vpc-032fdf526c5aa9a79 | weshopify-platform-vpc	10.0.10.0/24
weshopify-public-cloud-02	subnet-03a2cc986f76e7a14	Available	vpc-032fdf526c5aa9a79 | weshopify-platform-vpc	10.0.20.0/24

<u>Creating the Route Tables</u>

<u>Step-1:</u>

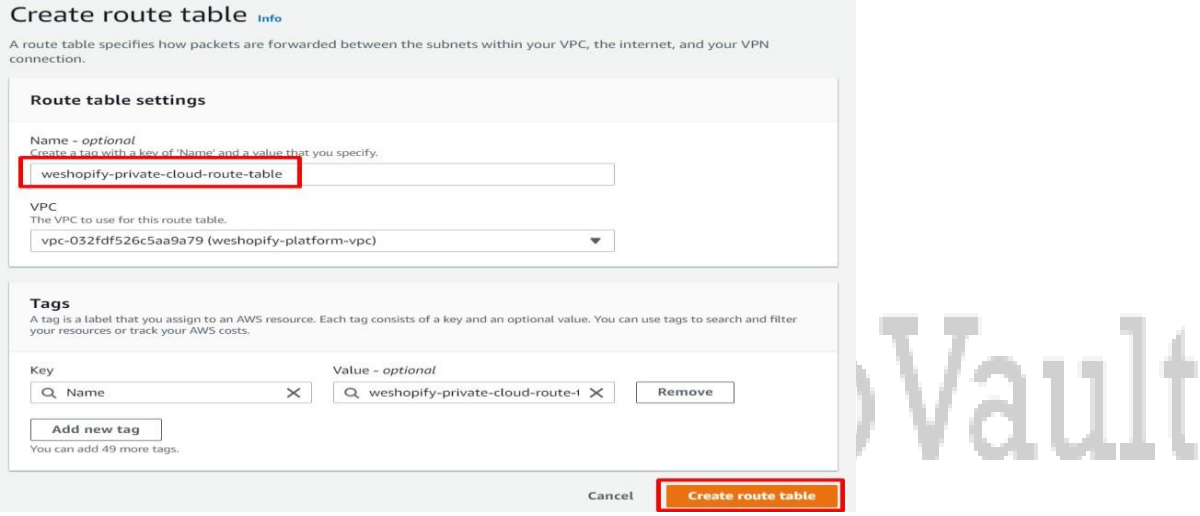
Create the 2 route tables later we can assign one for public cloud and another for private cloud . To Create the route tables click on the create route table button as shown below:



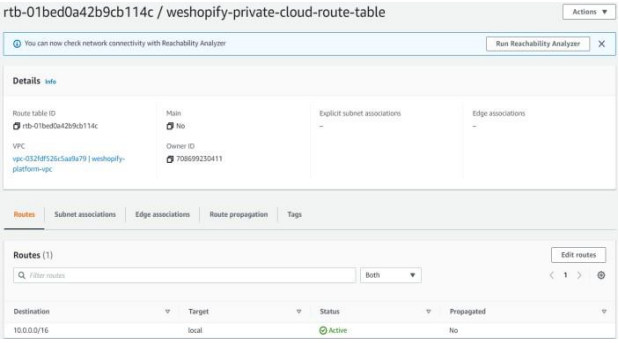
Note: here in the above screen shot we can see a route table created when our weshopify-platform-vpc created. We need to replace this default route table with the one we are now going to create.

Step-2: Create Route Table for private cloud

1. When we click on the create route table button as shown in the above screen shot we will get the below screen to enter the route names and then to create it

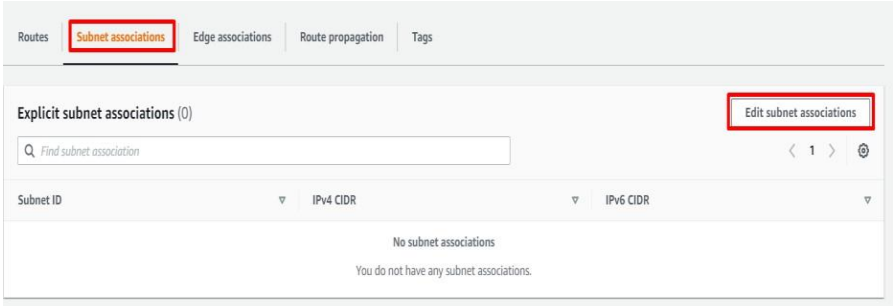


As soon as we click on create route table the route table will be created as shown below:

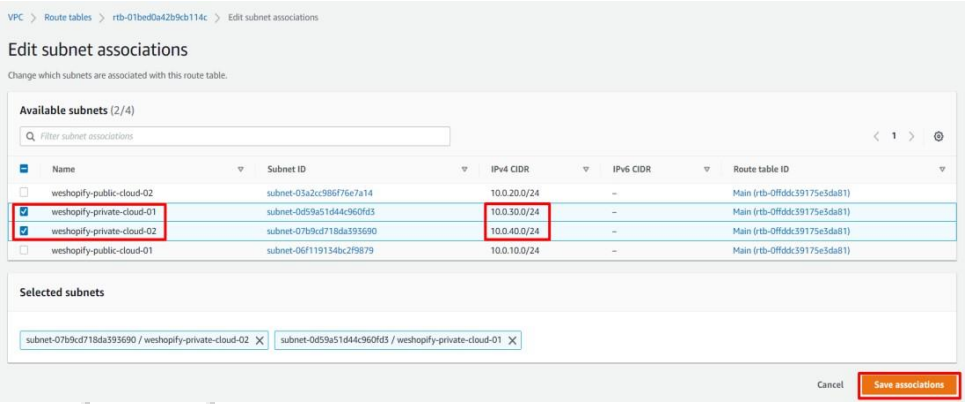


2. adding the subnet associations in the private cloud route as shown below:

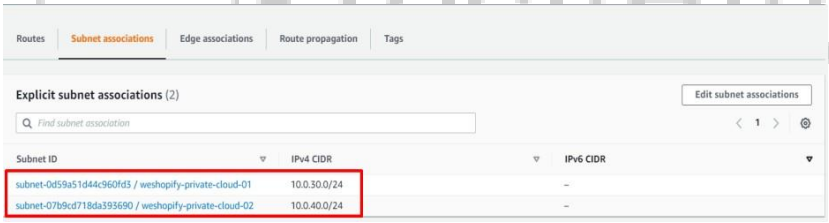
Click on Subnet associations and edit subnet associations as shown below:



3. as soon as we click on edit subnet associations the below screen will be opened for us to choose the subnets.

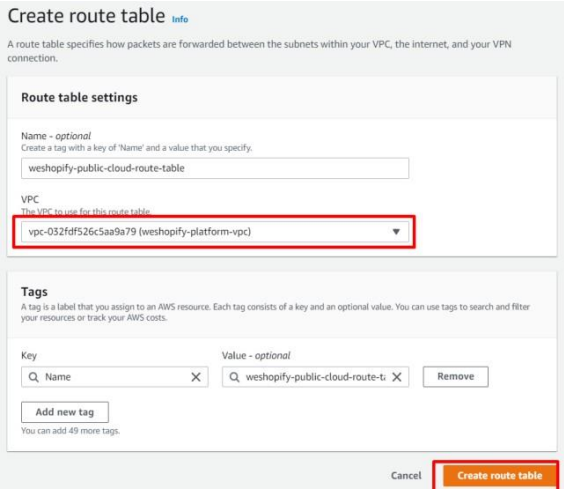


4. as soon as we click on save associations we can see the private subnets mapped to the private route table as shown below:

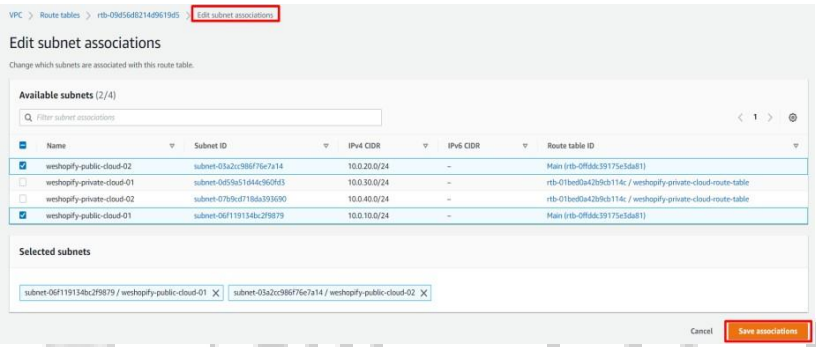


Step-3: Create Route Table for public cloud

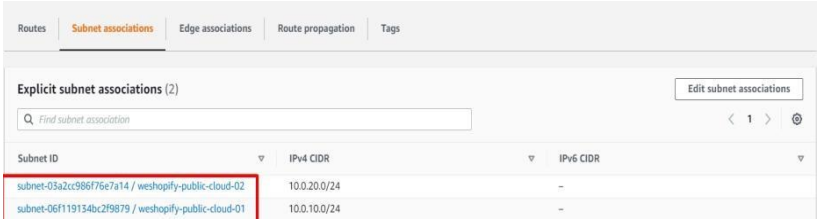
1. Create route table for public cloud as showb below



2. Associate public subnets using the public route table

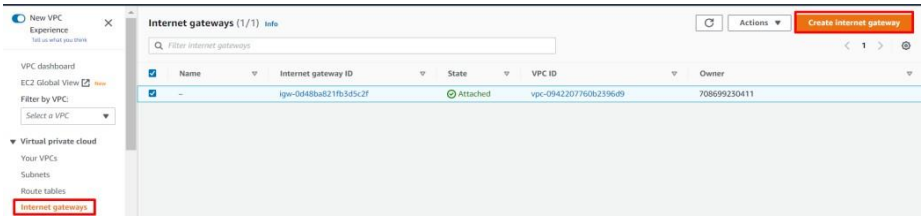


3. As soon as we click on the save associations we can see the public cloud subnets assocaited with the public cloud route table as shown below:

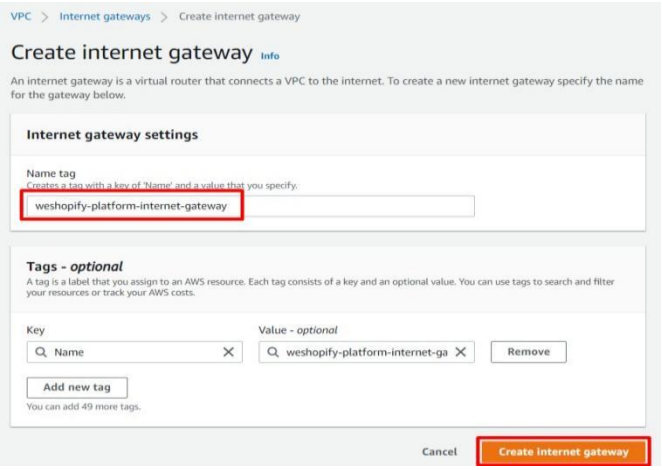


Creating the Internet Gateway

Step-1: To Create internet gateway, clcik on the Create Internet Gateway button as shown below:

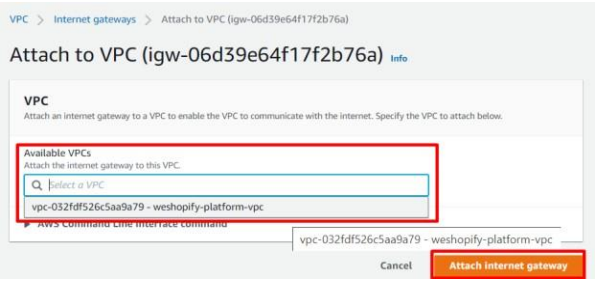


Step-2: as soon as we click on Create Internet Gateway we will get the below screen to ennter the name of thegateway

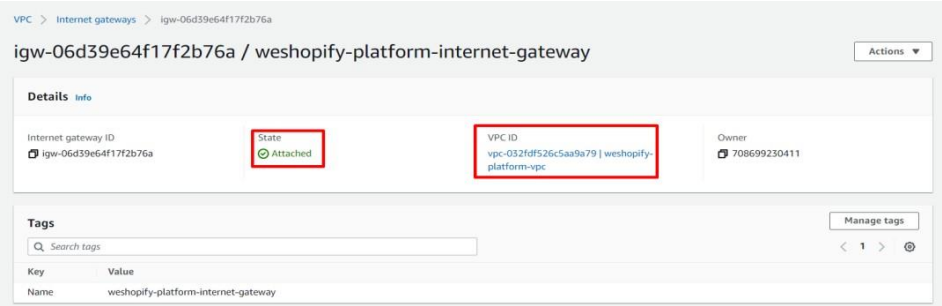


Step-3: once internetgateway created, attach it with the VPC as shown below:

As soon as we click on the Attach to VPC, the below screen shown to attach it to the available VPC's of our choise.

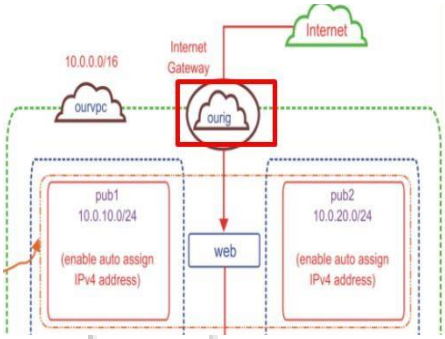


Step-4: Once the Internetgateway attached then we can see the internetgateway as shown below:

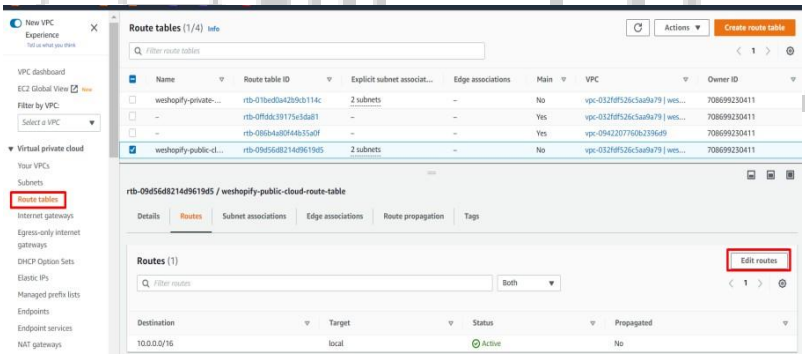


Adding the Internet Gateway to the public Route

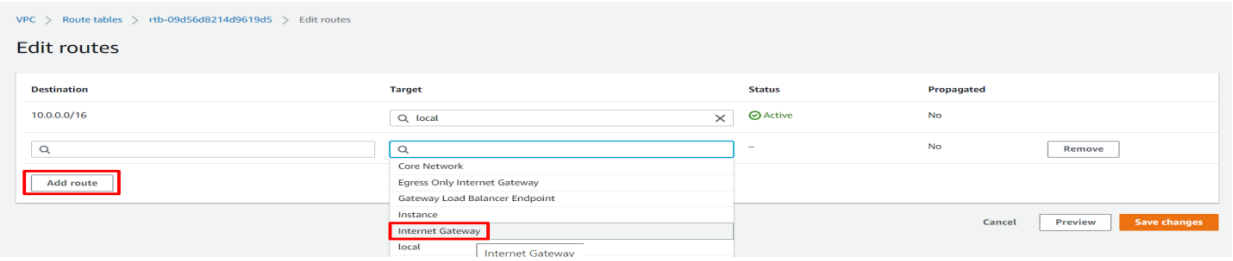
We have to attach the internet gateway to the public cloud subnets only to acces the resources those were deployed on the public cloud.

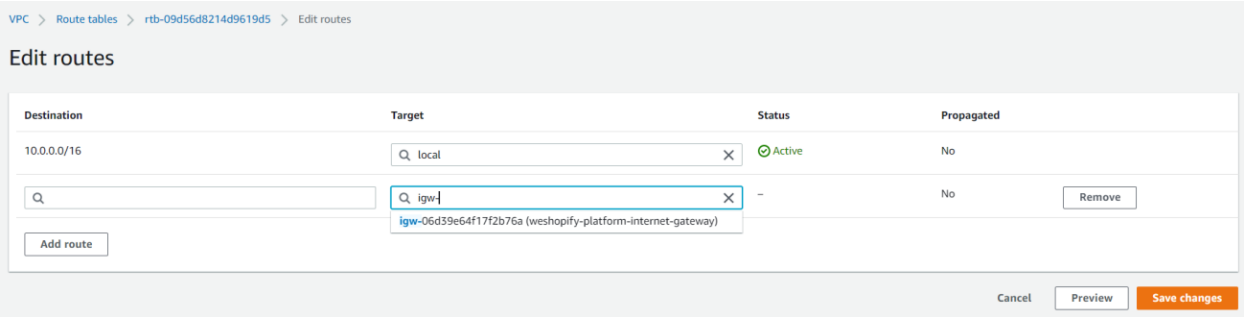


Goto Route tables and choose the public route table created as shown below:

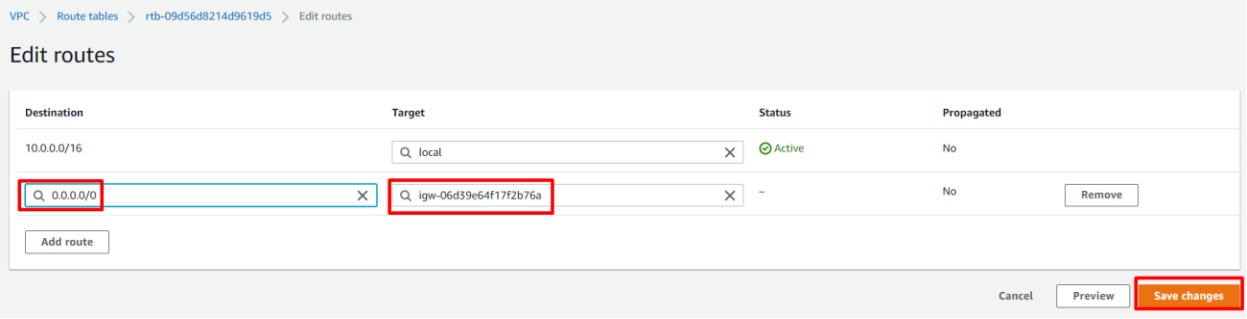


Click on Edit Route tables and attach the internet gateway

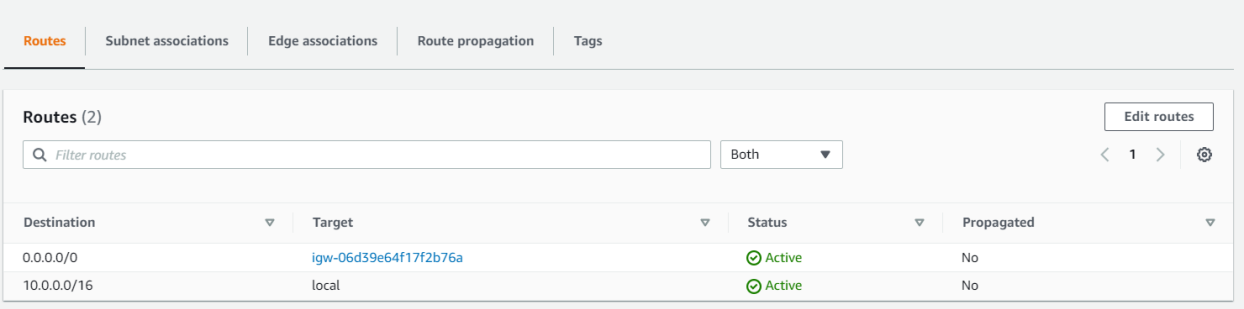




Configure from anywhere over the internet establish for our internet gateway



Now once we click on the save changes, we can see the below screen where it allows only the public subnets over the internet through the internet gateway.

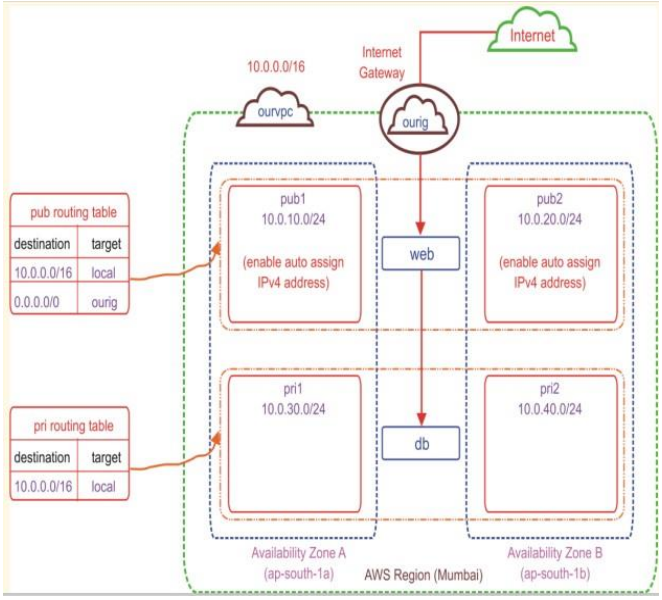


With these configurations I.e.

- ✓ VPC
- ✓ Subnets(private and public)
- ✓ Route table
- ✓ Internetgateway

We are now able to achieve the below VPC architecture.

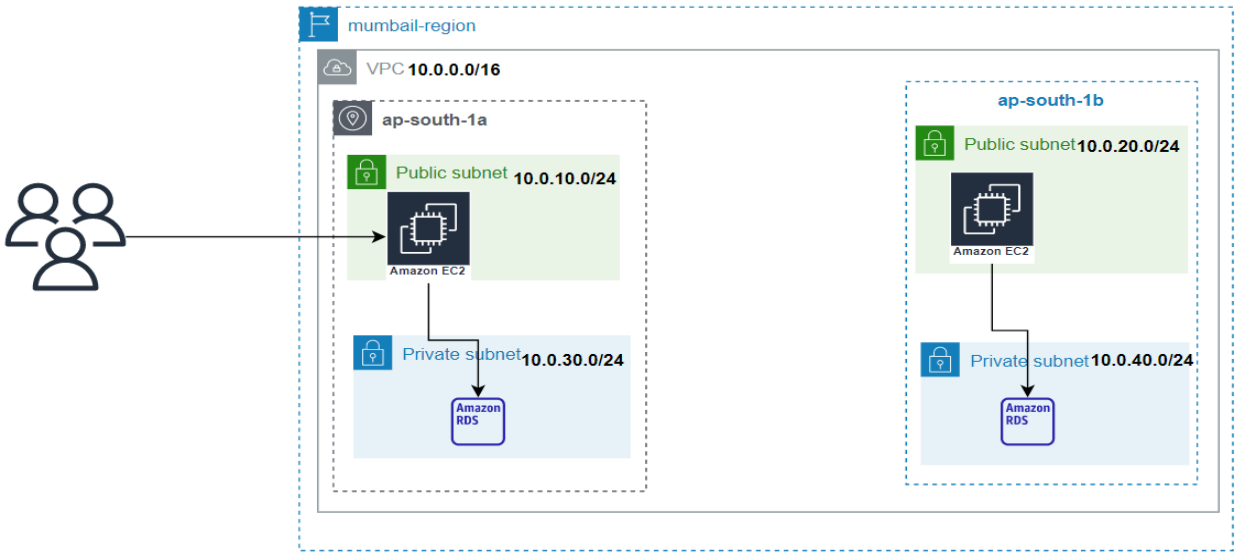
Amazon Virtual Private Cloud (Amazon VPC)



Application

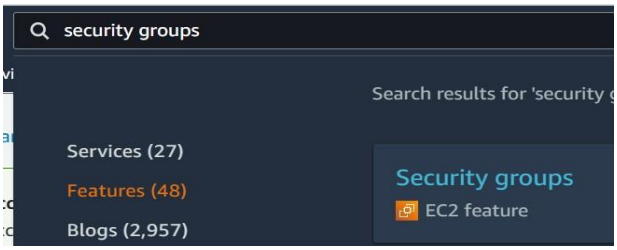
Use Case:

- ✓ Take one spring boot java application which is requires a a database connection. Take the ec2-machine in the public cloud and deploy the spring boot application in public cloud.
- ✓ Create Mysql database using **RDS** in private subnete
- ✓ Configure the mysql details in the spring boot application
- ✓ Now spring boot application should be able to connect to the database and the application can be accessible over the internet but not the database.

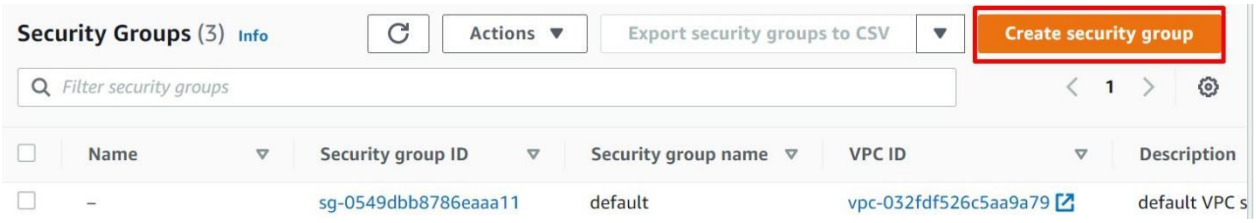


Create Security Group in our Custom VPC

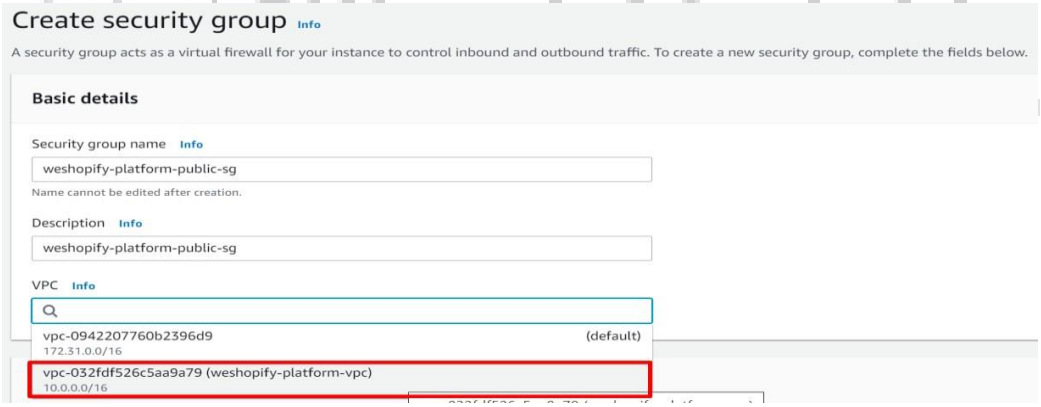
Step-1: To Create the security group search for security groups in search bar as shown below:



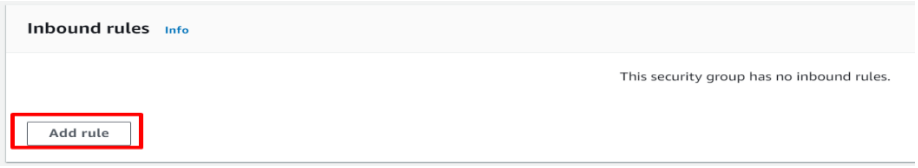
Step-2: as soon as we click on the security groups option as shown above we will get the following screen to create the security group



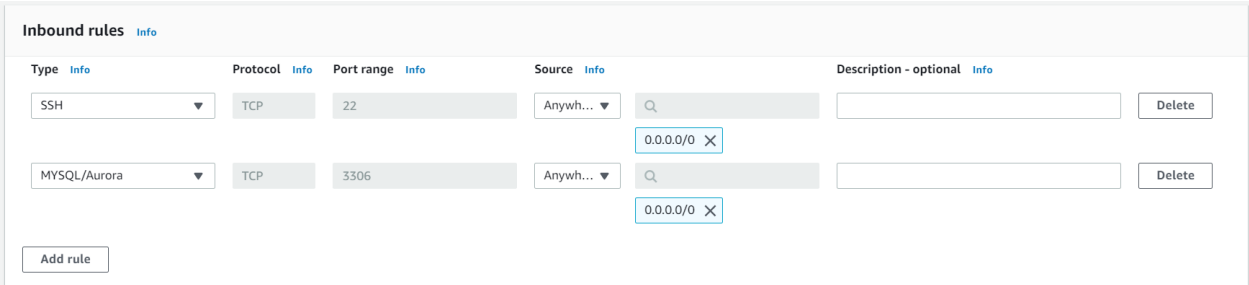
Step-3: as soon as we click on Create security group button as shown above the following screen will be opened to configure the security group in our VPC



Step-4: Configure the inbound rules, by clicking on the add rule button as shown below:

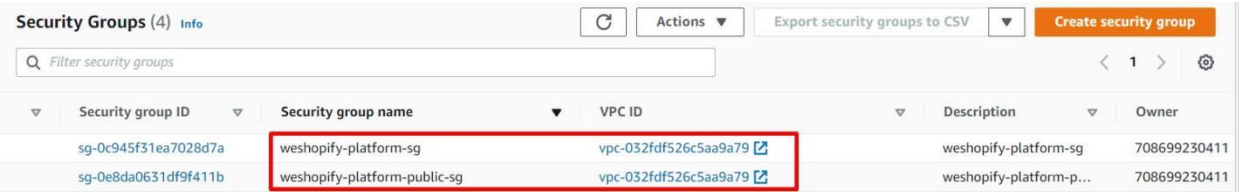


Step-5: Configure the inboud rules for SSH and for Mysql DB access as shown below:



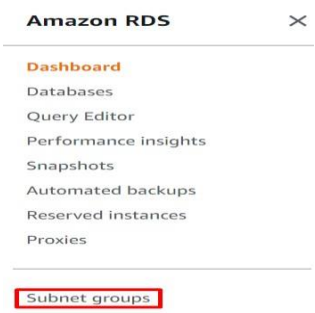
Step-6: clcik on create security group button once after done the above confogurations

Step-7: we can see the security group that we created in our custom VPC as shown below:

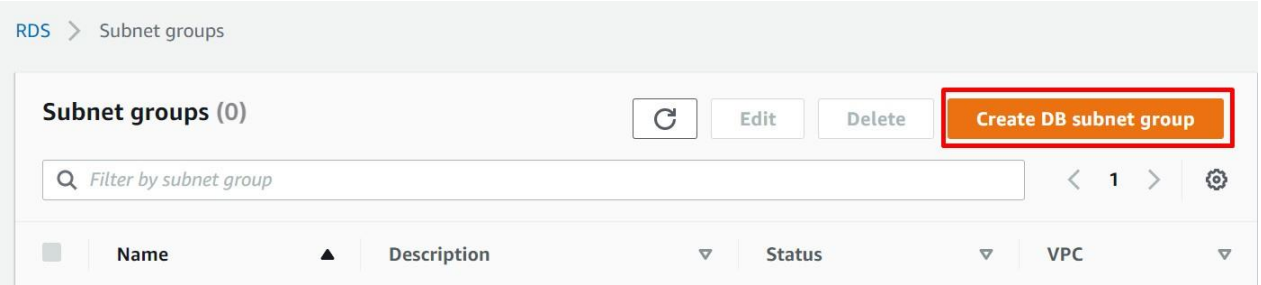


Create Mysql Database using RDS Service in Private Subnets

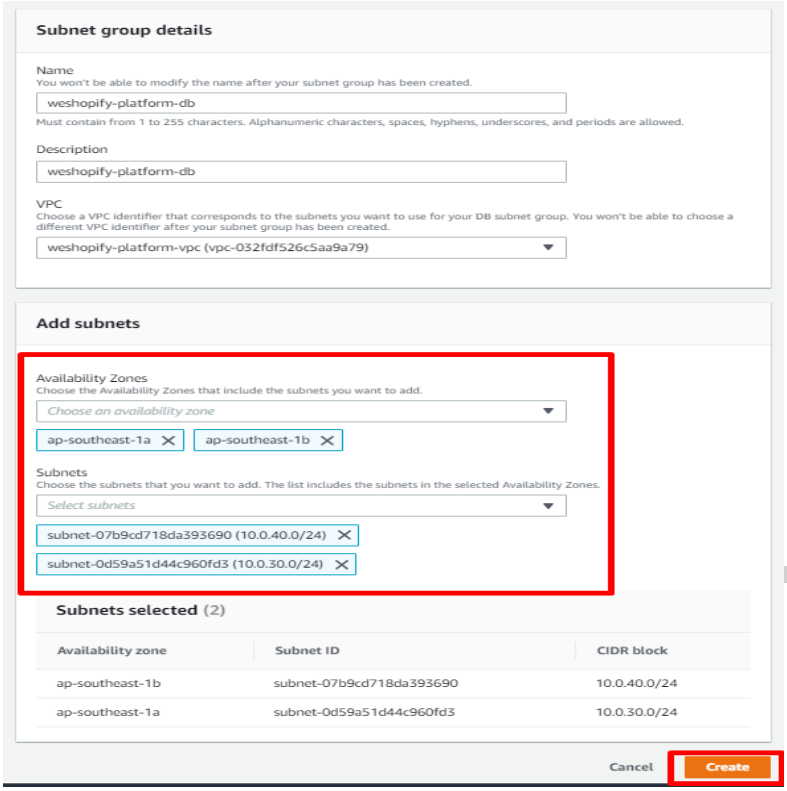
Step-1: Create database subnet groups, to configure the database in the private subnets as shown below:



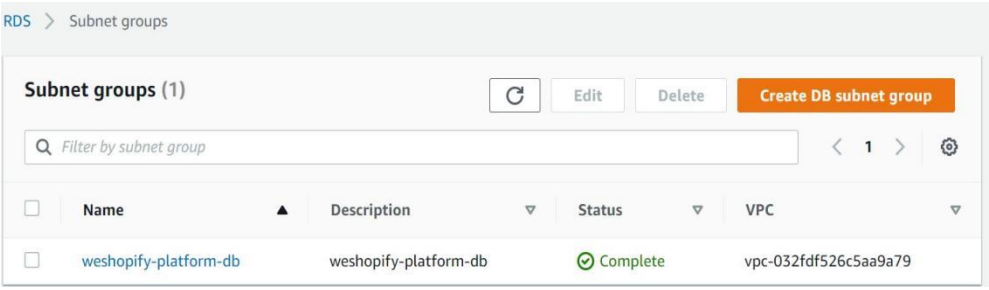
Step-2: Once we click on subnet groups, the following window will be opened to create the “DB Subnet Group” as shown below:



Step-3: Once we click on the Create DB subnet group button, the following screen will be opened where we can enter the name for the db subnet group, and choose the VPC, and region in which the subnets were created and finally choose the private subnets.

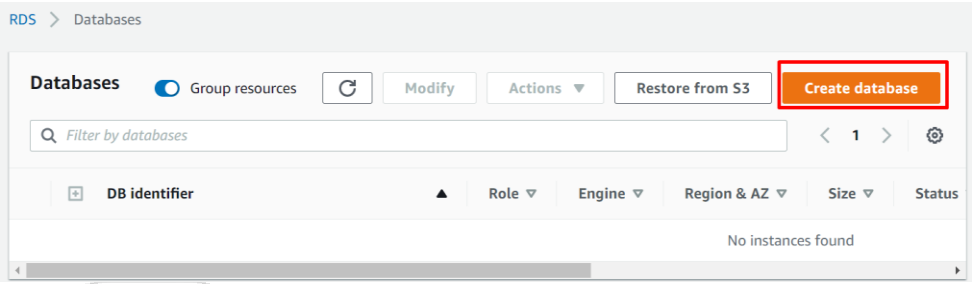


Step-4: Once we click on create button then we can see the db subnet group as shown below:



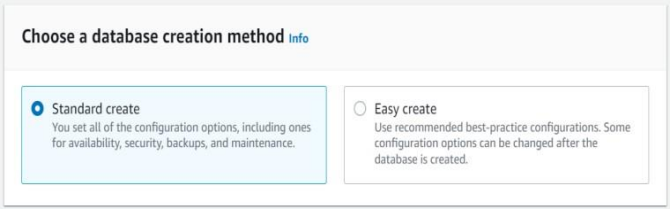
Step-5: Once the DB subnet group is created as shown above, lets create the DB now by choosing the databases option as shown below:

Step-6: once we click on the Databases options as shown above, we will get the below screen to create the Database.

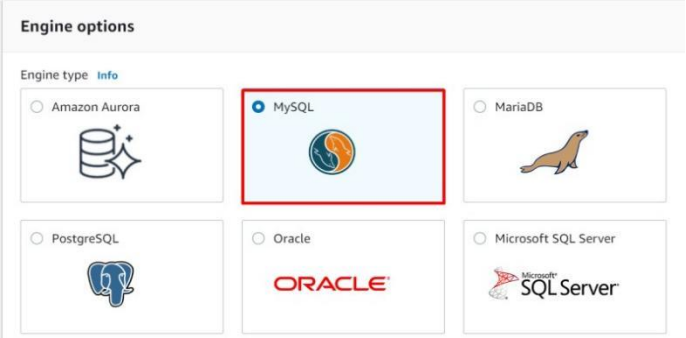


Step-7: Once we click on Create Database button as shown above, we will get the below screen:

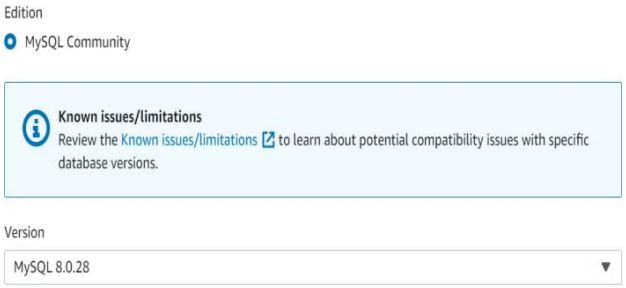
1. Choose a database



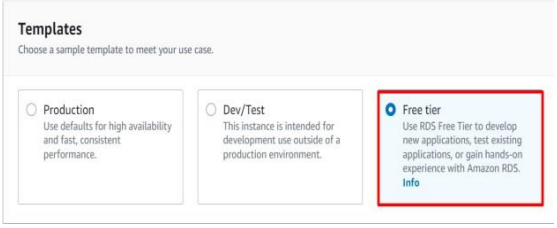
2. Select mysql



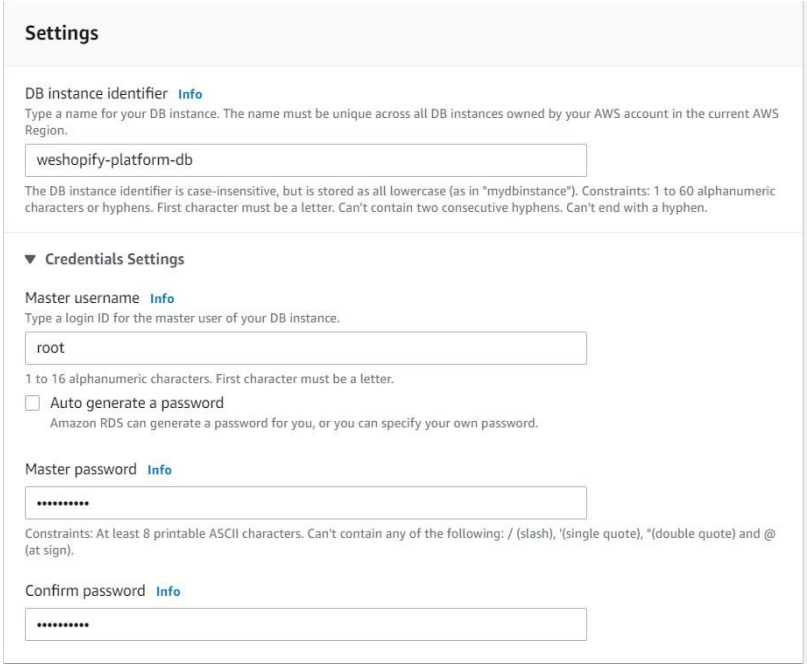
3. Choose ever latest mysql version from the list of available db versions



4. Choose the Free tier mysql instance



5. Configure the db settings like db name, user name, password as shown below:



Here I have choosen the password as : **Adance123$**

6. Choose the our VPC in the connectivity window:

Connectivity Info

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

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

Virtual private cloud (VPC) Info
Choose the VPC. The VPC defines the virtual networking environment for this DB instance.
weshopify-platform-vpc (vpc-032fdf526c5aa9a79)
Only VPCs with a corresponding DB subnet group are listed.

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

DB Subnet group Info
Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.
weshopify-platform-db

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

VPC security group (firewall) Info
Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.

- (●) Choose existing
  Choose existing VPC security groups
- ( ) Create new
  Create new VPC security group

Existing VPC security groups
Choose one or more options
default ✕ weshopify-platform-sg ✕

Availability Zone Info
No preference

7. Additioanl Configurations

Additional configuration
Database options, encryption turned off, backup turned off, backtrack turned off, maintenance, CloudWatch Logs, delete protection turned off.

Database options

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

DB parameter group Info
default.mysql8.0

Option group Info
default:mysql-8-0

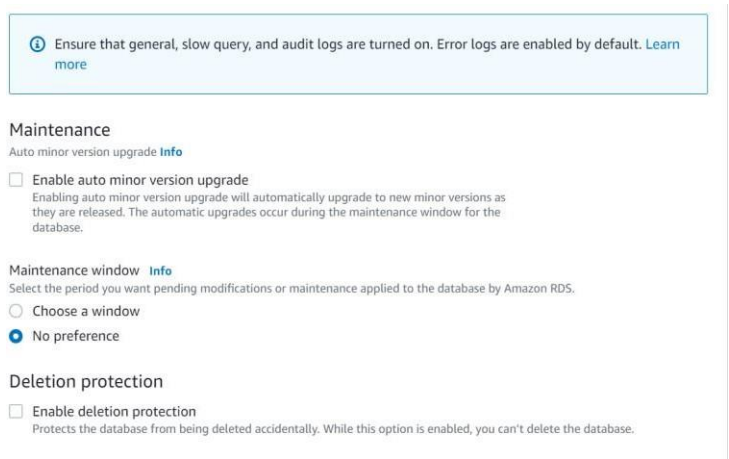
[ ] Enable automated backups
Creates a point-in-time snapshot of your database

Encryption

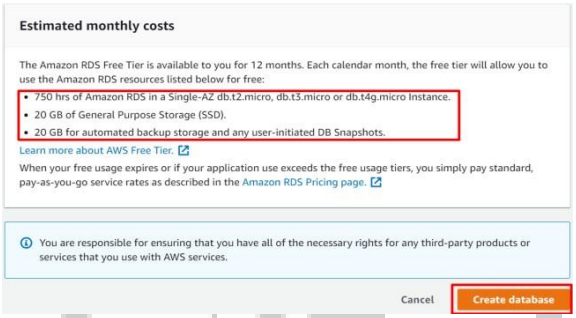
[ ] Enable encryption
Choose to encrypt the given instance. Master key IDs and aliases appear in the list after they have been created using the AWS Key Management Service console. Info

Log exports
Select the log types to publish to Amazon CloudWatch Logs
[ ] Audit log
[ ] Error log
[ ] General log
[ ] Slow query log

IAM role
The following service-linked role is used for publishing logs to CloudWatch Logs.
RDS service-linked role

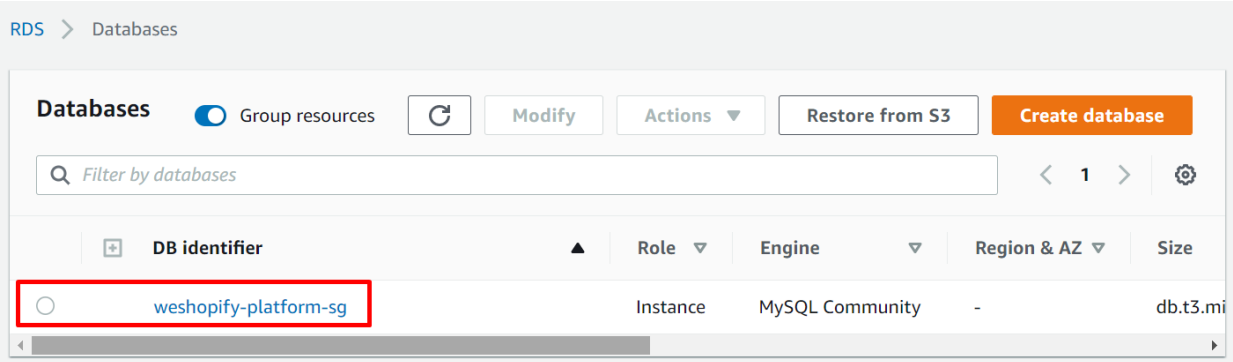


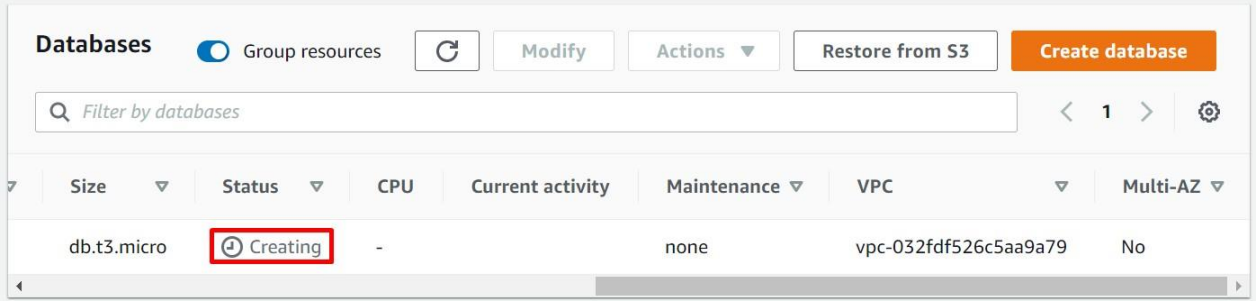
Note: here in the above screen, the **"Initial database name"** will represents the name of the database schema, in which the tables will be created



Finally click on Create Database button.

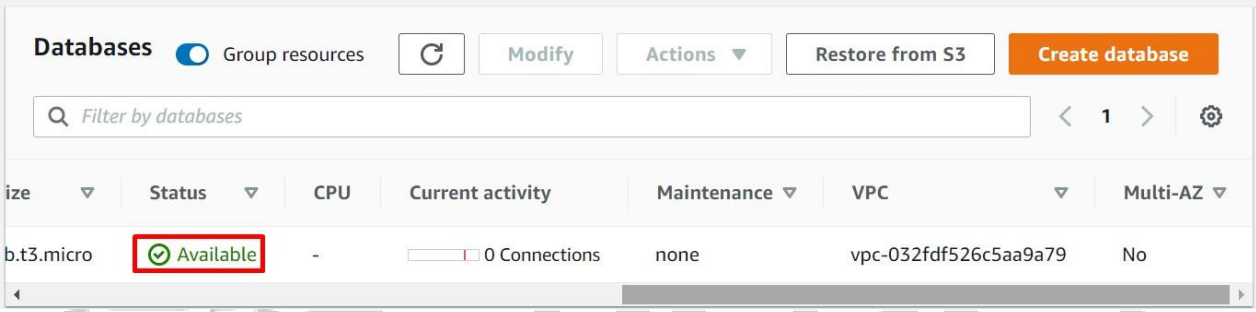
8. As soon as we click on Create Database button, we can see the data base creating as shown below.





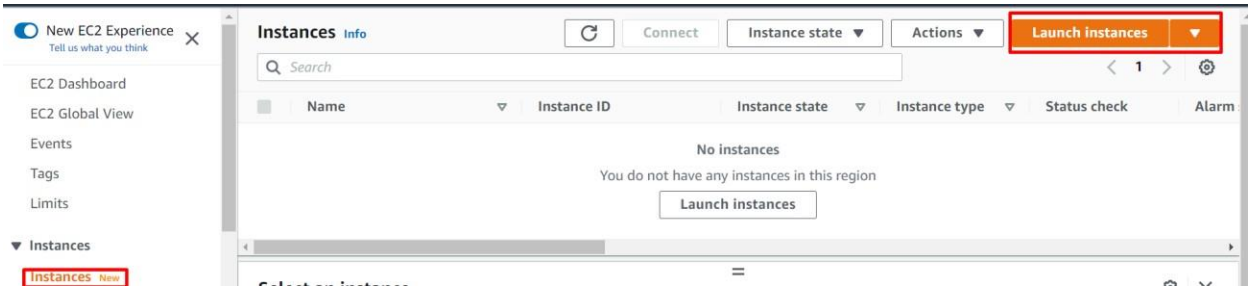
This will take little longer time, but once came to created status, we can connect to it from the AWS resource like EC2 using the command line utility but we cant access it over internet as we have configured it in the private subnets.

Once the database is available after around 10-15 mins of time we can see its status as available as shown below:



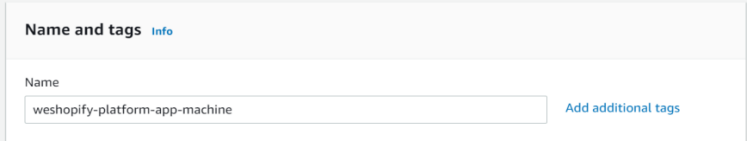
Create Ec2 instance in Public Subnets

Step-1: To Launch EC2 instances, clcik on Instances link as shown below

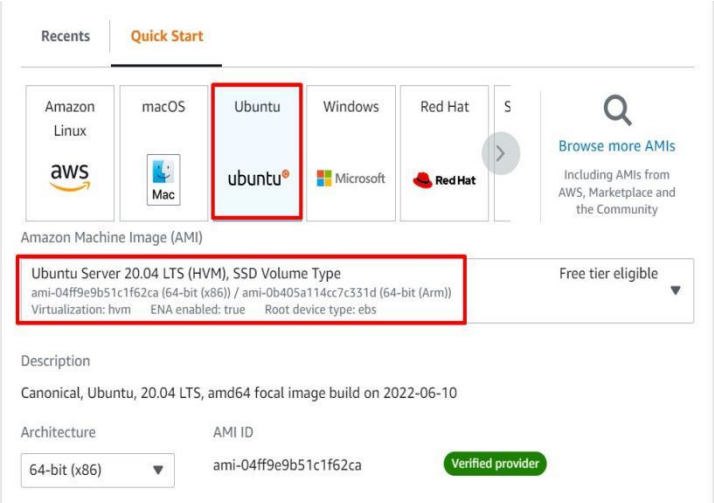


Step-2: Click on Launch Instances button as shown above, to create the instance

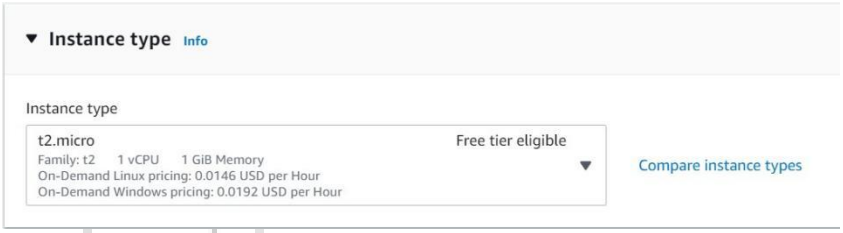
1. Name and Tags



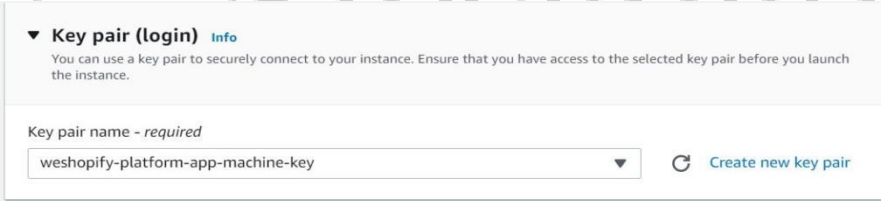
2. OS Choice



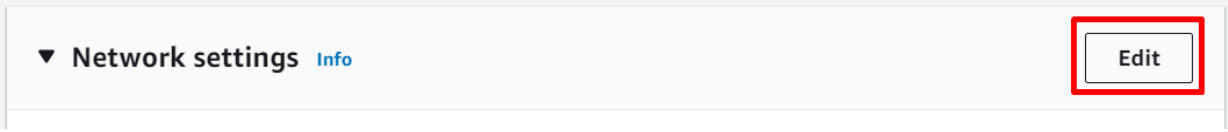
3. Instance Type

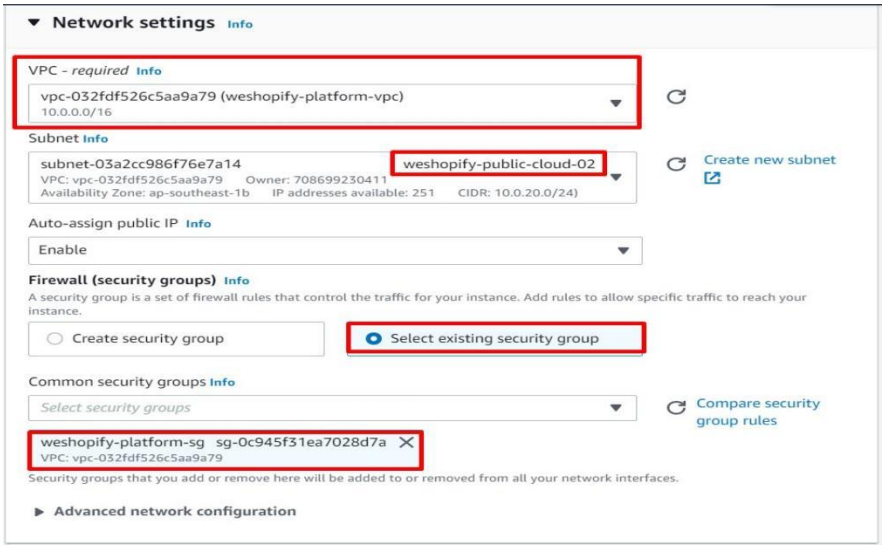


4. Choose the key pair if already created other wise click on create key pair

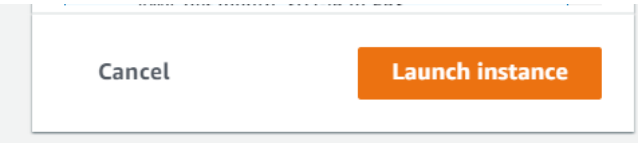


5. Edit Network settings as shown below to configure this ec2 machine in our vpc as shown below



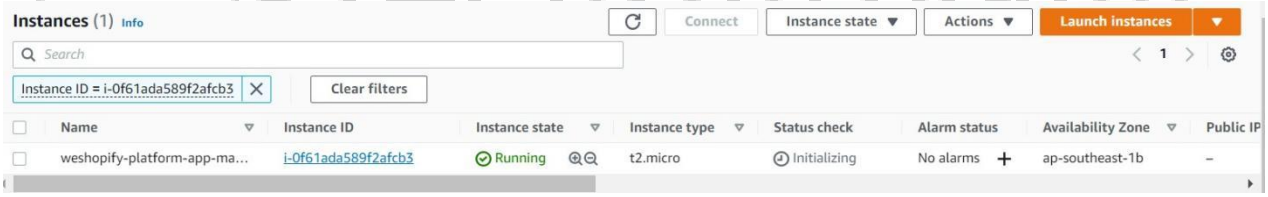


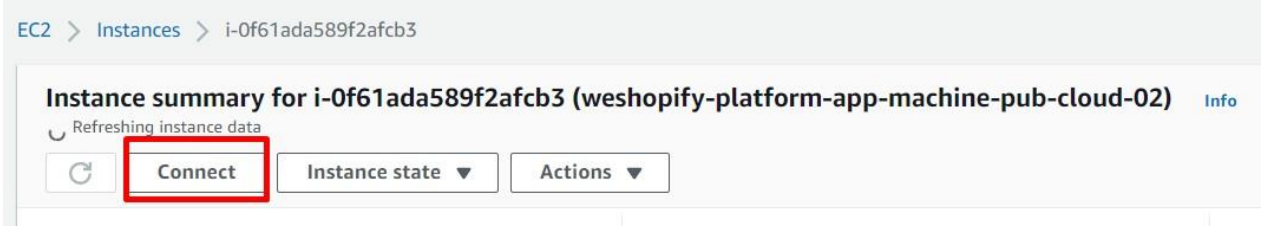
Clcik on launch instance as shown below:



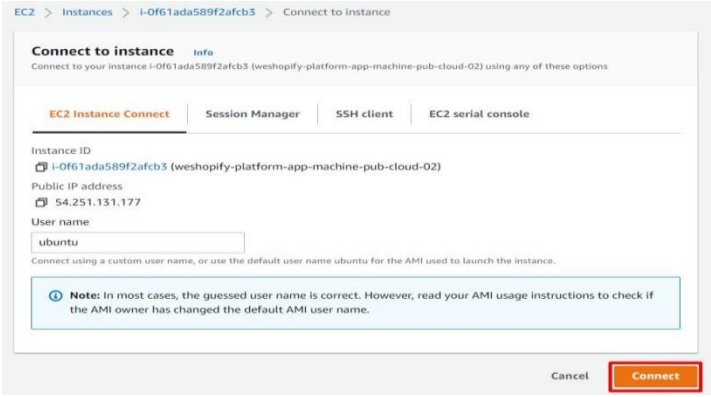
Step-2: Connect to RDS from the ec2 instance that we created

1. To Connect to the DB from the ec2 instance, goto the ec2 instances list and select the instance what we created and click on connect button as shown below:

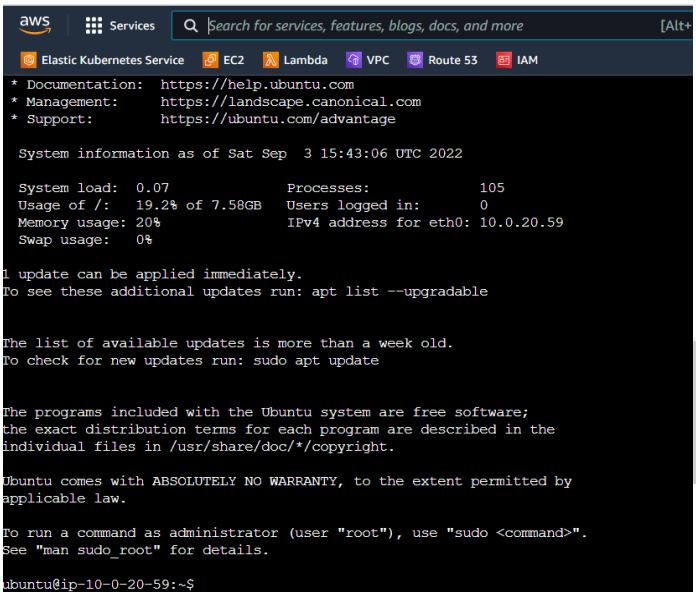




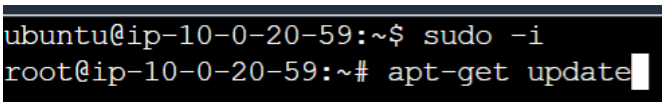
2. Once we click on connect the following window will be open clcik on connect.



3. As soon as we click on connect the ec2 machine will be opend in the browser as shown below:



4. Switch to root user and update the machine



5. Install the mysql client as shown below

```
root@ip-10-0-20-59:~# mysql

Command 'mysql' not found, but can be installed with:

apt install mysql-client-core-8.0     # version 8.0.30-0ubuntu0.20.04.2, or
apt install mariadb-client-core-10.3  # version 1:10.3.34-0ubuntu0.20.04.1

root@ip-10-0-20-59:~# apt install mysql-client-core-8.0
```

6. Once the mysql client will install we can connect to the RDS from the ec2 instance using the below command:

```
root@ip-10-0-20-59:~# mysql -h weshopify-platform-sg.cci53w9t6upc.ap-southeast-1.rds.amazonaws.com -P 3306 -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 23
Server version: 8.0.28 Source distribution

Copyright (c) 2000, 2022, 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>
```

Command is: mysql -h <RDS_MYSQL_ENDPOINT> -P 3306 -u <user_name> -p

For example: mysql -h weshopify-platform-sg.cci53w9t6upc.ap-southeast-1.rds.amazonaws.com -P 3306 -u root -p

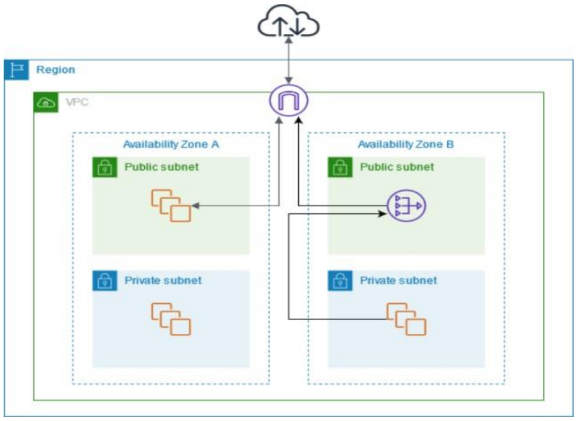
Here the RDS_MYSQL_ENDPOINT can be collected from the below screen:

NAT Gateway

Lets run the application inside the ec2 machine which we will set up in private subnet. But the application which is now in private subnet cant be accessed by the users over the internet. So the private network needs to be translate to the public network. To do this we will use the NAT Gateway.

The NAT gateway sends the traffic to the internet gateway, using its Elastic IP address as the source IP address.

Amazon Virtual Private Cloud (Amazon VPC)



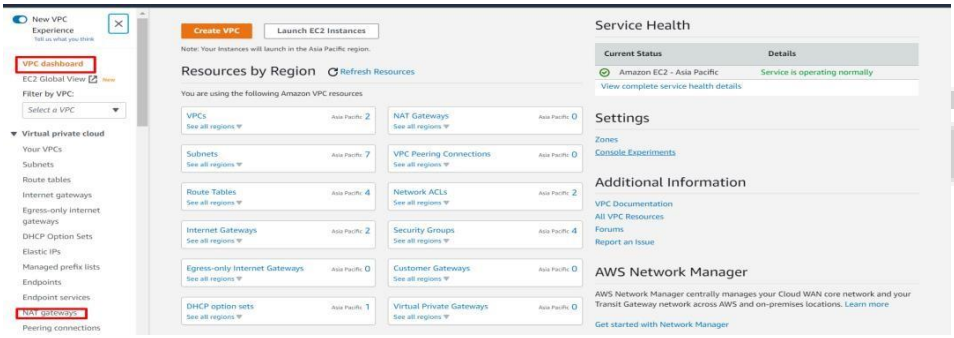
Here

 means NAT Gateway

 means Internet gateway

NAT Gateway Setup

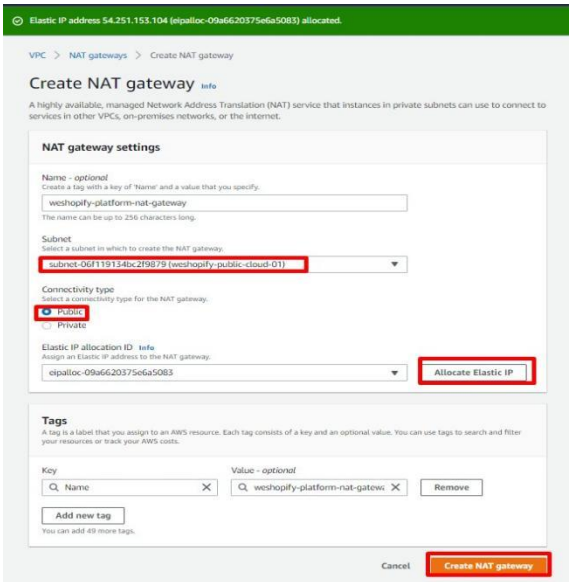
Step-1: To setup the NAT Gateway, click on the NAT Gateway option as shown below:



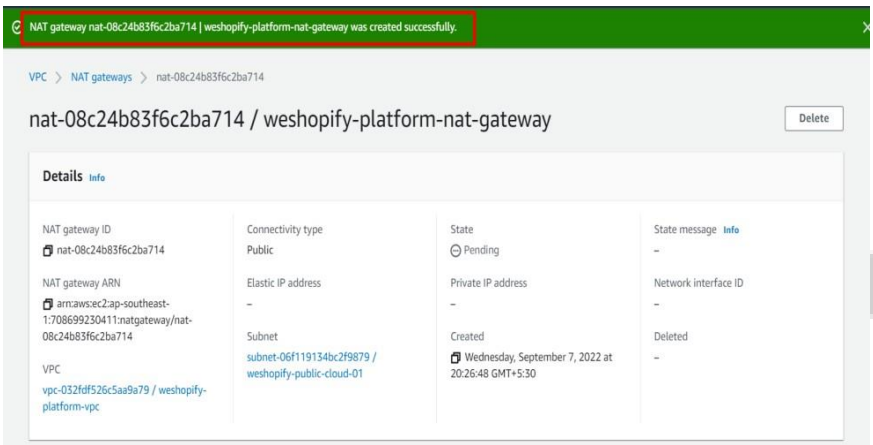
Step-2:

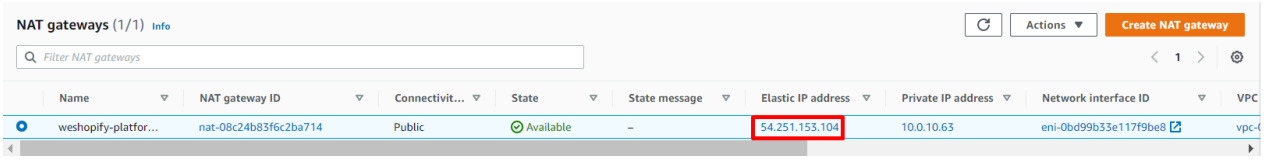
Once we click on the Nat gateways, then the screen will be opened like as below:

Step-3: when we clcik on Create NAT Gateway, then the following screen will be opened where we can enter the name of the nat gateway and choose the subnet. Note that always **NAT gateway should be set up in public subnet only.**



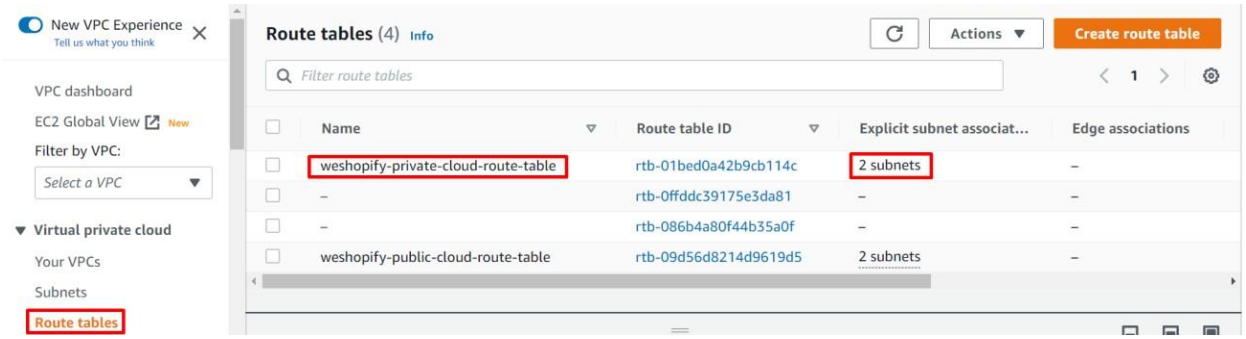
Step-4: Once we clcik on Create NAT gateway, then we will see the NAT Gateway created as shown below:



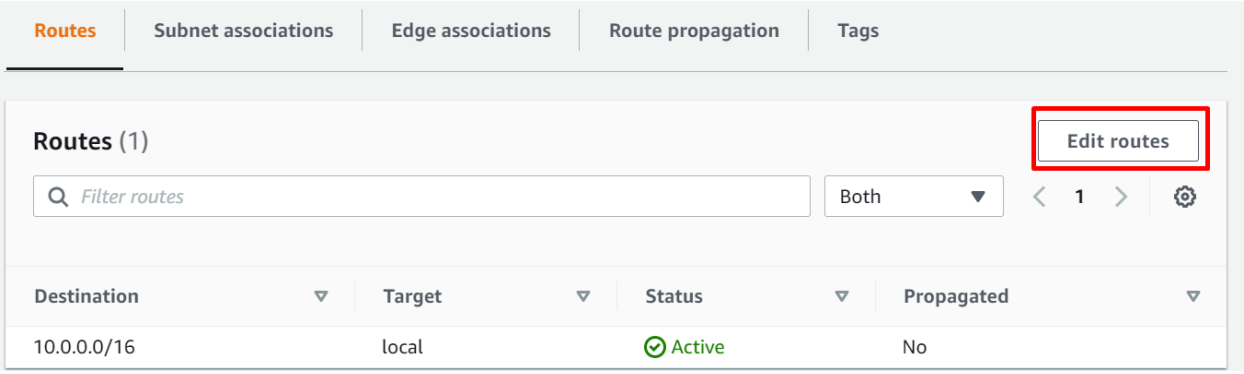


Attach NAT Gateway to the private subnet

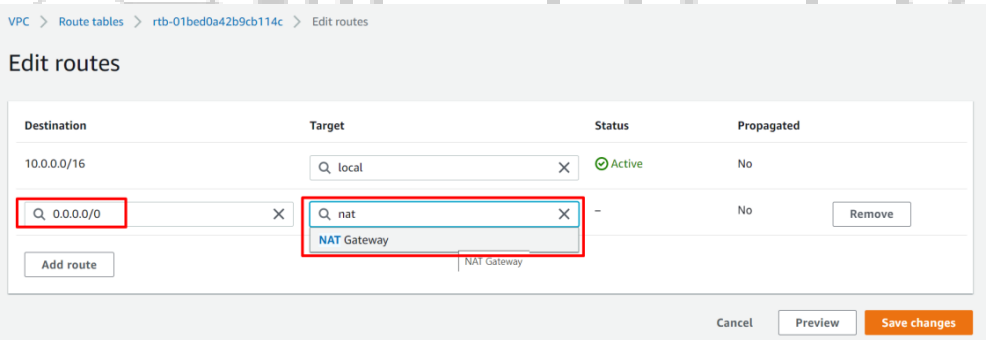
Step-1: go to route tables as shown below and choose the private route table in which we would like to attach the NAT Gateway.

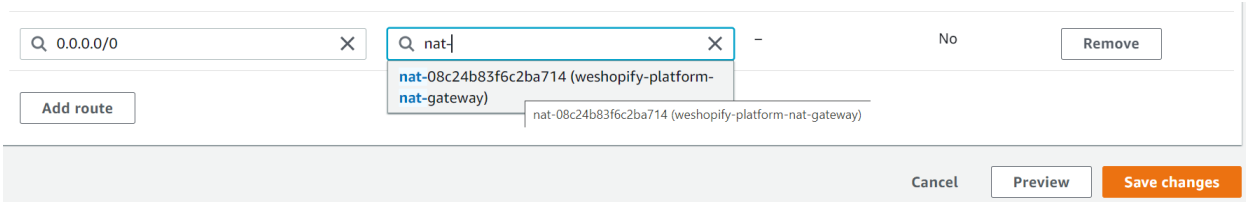


Step-2: Once we click on private cloud-01 route table, select edit routes



Step-3: once we click on Edit routes, the following screen will be opened where we can add the route for the NAT Gateway as shown below:



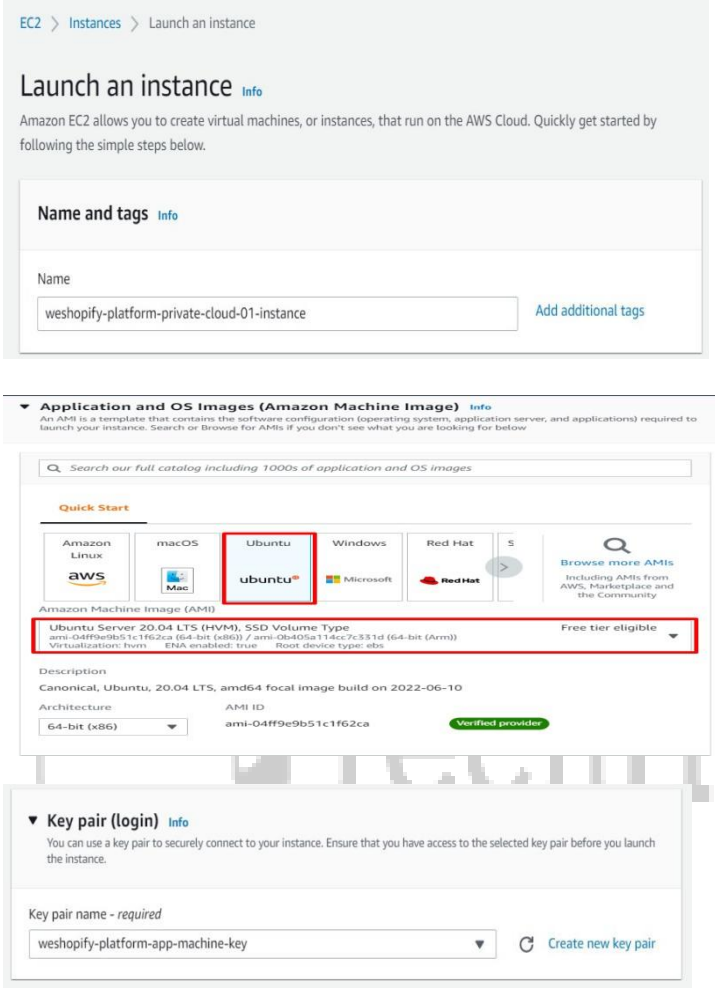


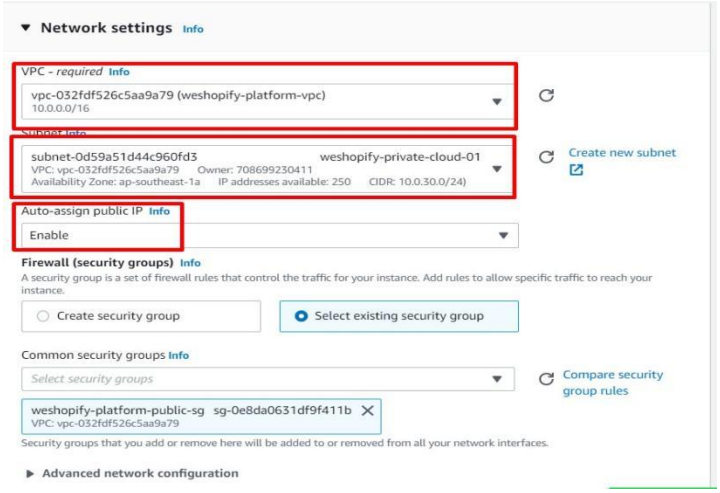
Click on save changes.

Testing NAT Gateway Connection

Step-1: Create an EC2 machine in the private subnet

To Test NAT Gateway connection, we have to create an ec2 machine in private subnet as shown below, as we have attached the NAT Gateway to the private subnet's route table.





Note

Please note a small correction in the above screen shot I.e. Auto-assign public IP choosen as Enable but it must be as “Disable”. because we should not connect to the ec2 machine in the private cloud from an external over the internet.

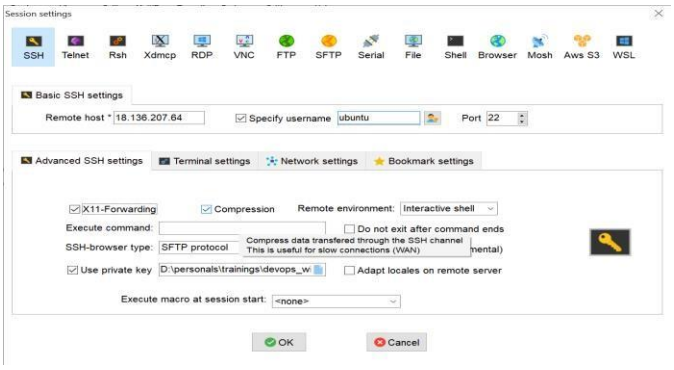
Clcik on laucnh instances

Step-2: Create an EC2 machine in the public subnet

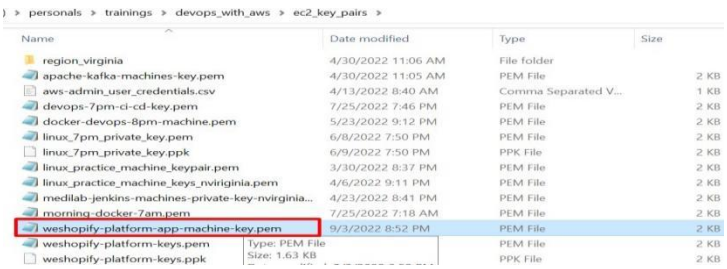
Repeat the same steps as above but while editing the networking settings, choose the public subnet.

Step-3: Connect the private cloud ec2 machine from the public cloud ec2 machine

First connect to public machine using the moba xterm or directly AWS Connect button over the web.



When we have created the ec2 machine in private cloud , we must have downloaded the key to connect to the machine as shown below:



On public coud machine, upload create a pem file with the content of the above file or otherwise upload this pem file as shown below:

```
root@ip-10-0-20-59:~# ls -l
total 8
-rw------- 1 root root 1679 Sep  8 15:24 privatemachine.pem
drwx------ 4 root root 4096 Sep  3 15:40 snap
root@ip-10-0-20-59:~#
```

Once after the file is created I.e. like above privatemachine.pem on the public machine connect to the private machine, change its permission to **"chmod 600 privatemachine.pem"**

This means providing the read and write permissions on the file to only the owner who created it.

Once after the file is uploaded/created, lets use the below command to connect to the private cloud machine from public cloud machine.

Remember as the both the machines are in same VPC I.e. our custom VPC, the machines able to communicate each other using the private ip address.

Command: ssh -i <pem_file_of_private_cloud_machine> <user_name>@<private_ip>

ssh -i privatemachine.pem ubuntu@10.0.30.194

```
root@ip-10-0-20-59:~# ls -l
total 8
-rw------- 1 root root 1679 Sep  8 15:24 privatemachine.pem
drwx------ 4 root root 4096 Sep  3 15:40 snap
root@ip-10-0-20-59:~# ssh -i privatemachine.pem ubuntu@10.0.30.194
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.15.0-1019-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat Sep 10 15:05:04 UTC 2022

  System load:  0.0                Processes:             101
  Usage of /:   27.6% of 7.57GB    Users logged in:       1
  Memory usage: 20%                IPv4 address for eth0: 10.0.30.194
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

24 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Sat Sep 10 15:02:24 2022 from 10.0.20.59
ubuntu@ip-10-0-30-194:~$
```

Set up the application on the private machine as shown below:

```
root@ip-10-0-30-194:/opt/weshopify-platform-monolith-app# ls -l
total 36
-rw-r--r-- 1 root root    33 Sep 10 15:15 README.md
-rw-r--r-- 1 root root 10284 Sep 10 15:15 mvnw
-rw-r--r-- 1 root root  6734 Sep 10 15:15 mvnw.cmd
-rw-r--r-- 1 root root  3493 Sep 10 15:15 pom.xml
drwxr-xr-x 4 root root  4096 Sep 10 15:15 src
drwxr-xr-x 9 root root  4096 Sep 10 15:20 target
root@ip-10-0-30-194:/opt/weshopify-platform-monolith-app# cd target/
root@ip-10-0-30-194:/opt/weshopify-platform-monolith-app/target# ls -l
total 88960
drwxr-xr-x 5 root root     4096 Sep 10 15:19 classes
drwxr-xr-x 3 root root     4096 Sep 10 15:19 generated-sources
drwxr-xr-x 3 root root     4096 Sep 10 15:19 generated-test-sources
drwxr-xr-x 2 root root     4096 Sep 10 15:19 maven-archiver
drwxr-xr-x 3 root root     4096 Sep 10 15:19 maven-status
drwxr-xr-x 3 root root     4096 Sep 10 15:19 test-classes
drwxr-xr-x 4 root root     4096 Sep 10 15:19 we-shopify-platform
-rw-r--r-- 1 root root 49497803 Sep 10 15:20 we-shopify-platform.war
-rw-r--r-- 1 root root 41562172 Sep 10 15:20 we-shopify-platform.war.original
root@ip-10-0-30-194:/opt/weshopify-platform-monolith-app/target#
root@ip-10-0-30-194:/opt/weshopify-platform-monolith-app/target# java -version
openjdk version "11.0.16" 2022-07-19
OpenJDK Runtime Environment (build 11.0.16+8-post-Ubuntu-0ubuntu120.04)
OpenJDK 64-Bit Server VM (build 11.0.16+8-post-Ubuntu-0ubuntu120.04, mixed mode, sharing)
root@ip-10-0-30-194:/opt/weshopify-platform-monolith-app/target#
root@ip-10-0-30-194:/opt/weshopify-platform-monolith-app/target# java -jar we-shopify-platform.war
```

Now login back to public machine in next window of the mobaxterm and do a curl to the private machine where the application is running as shown below then we should be able to access the application:

```
root@ip-10-0-20-59:~# curl 10.0.30.194:5001
<!doctype html>
<html lang="en">

<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <title>Login</title>
    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="../assets/vendor/bootstrap/css/bootstrap.min.css">
    <link href="../assets/vendor/fonts/circular-std/style.css" rel="stylesheet">
    <link rel="stylesheet" href="../assets/libs/css/style.css">
    <link rel="stylesheet" href="../assets/vendor/fonts/fontawesome/css/fontawesome-all.css">
    <style>
    html,
    body {
        height: 100%;
    }
```