Discipline

ownCloud using AWS

CS5002

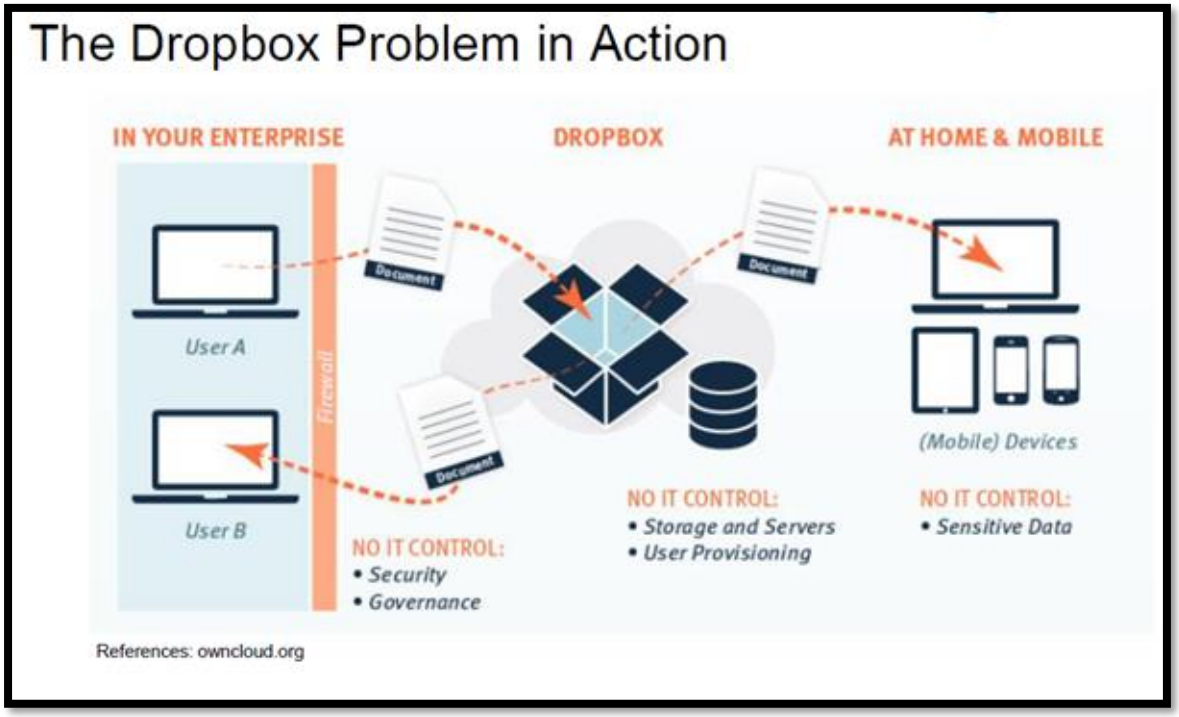
Course Project

Sheikh Muhammed Tadeeb (AU19B1014)

❖ Existing Problem:

According to recent research, 40-75% of employees are using Dropbox to share files inside and outside of their businesses. Half of those Dropbox users do this even though they know it's against the rules. More than 40% of businesses have experienced the exposure of confidential information and the estimated average cost of a data breach equalled $5.5 Million in 2011.

These files, containing sensitive company and customer data, are stored in a public cloud outside of the businesses' control - possibly even outside of the country. The potential for data leakage and security breaches is enormous and companies need to stay compliant with their own policies and procedures for security and governance.



❖ Problem Statement:

The objective of this project is to deploy infrastructure to deploy ownCloud application support with MYSQL database with the help of AWS capabilities.

❖ Objective:

The following are the objectives of this project:

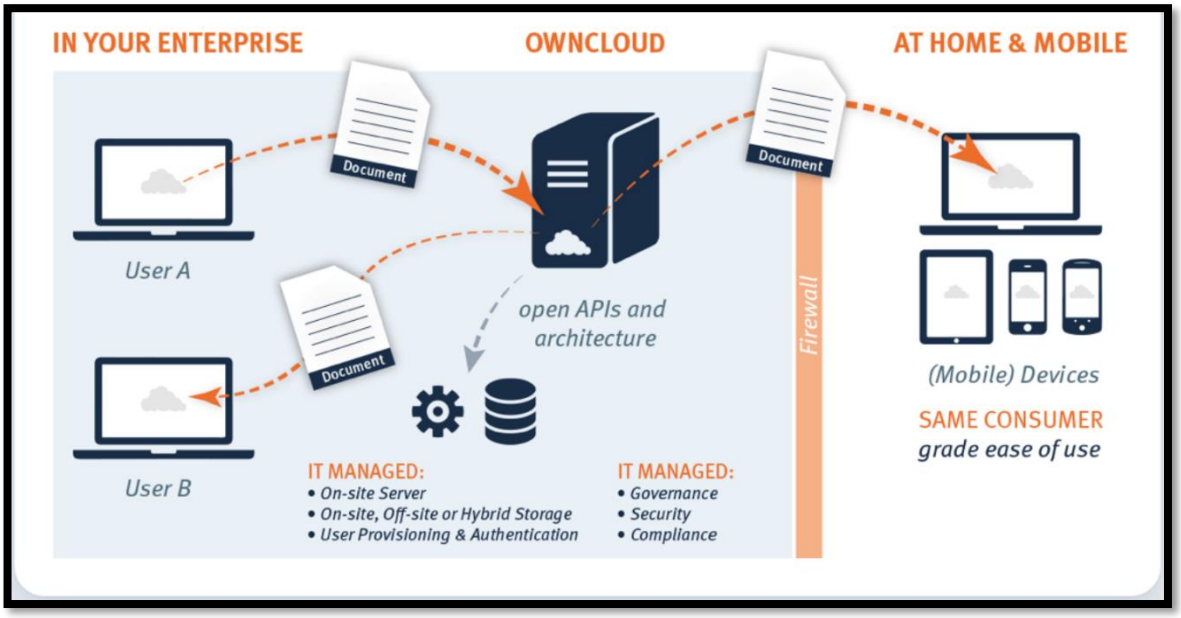
1. Implement the ownCloud solution for a small workgroup, which can cater up to 150 users by using various AWS services. The solution will be completely deployed on AWS. i.e., Implement 2 different subnets (one public and other private) in a custom VPC called owncloud-vpc.

2. The ownCloud app should be installed in public subnet and MUST be configured to access a new database called owncloud-db (created by you) in the private subnet.

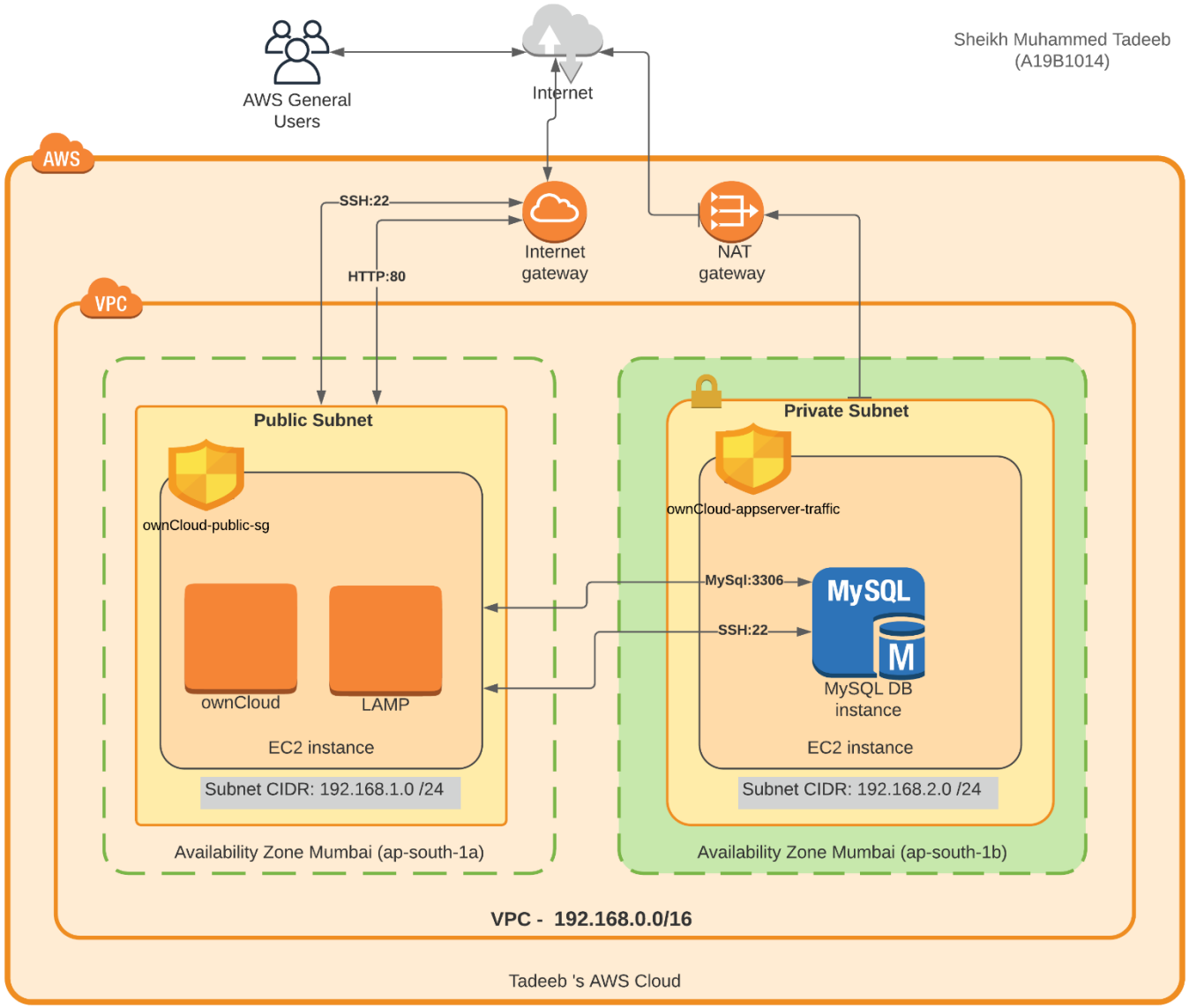
3. Apache HTTP server should host ownCloud application in this subnet and must be configured with required PHP modules for ownCloud.

❖ Proposed Solution:

ownCloud is an open-source secure file sync and share solution which can help you gain control of this situation and enable you to create and deploy an enterprise scale file solution. ownCloud can run in our data centre or on a public cloud, with its servers, storage etc completely managed and controlled by your IT team and management in accordance with our company's governance and security requirements. I have decided to launch the ownCloud service from AWS.

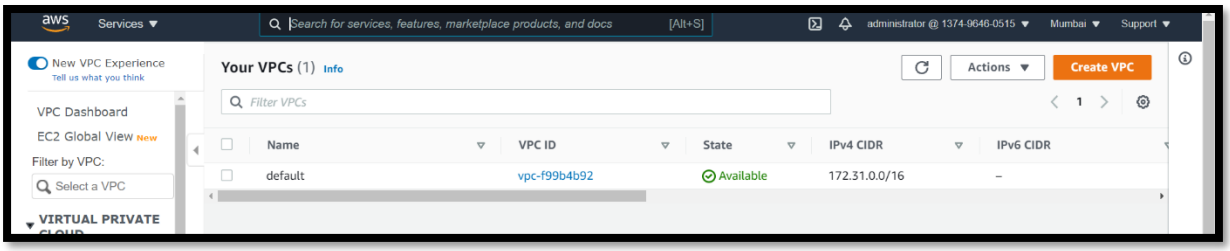


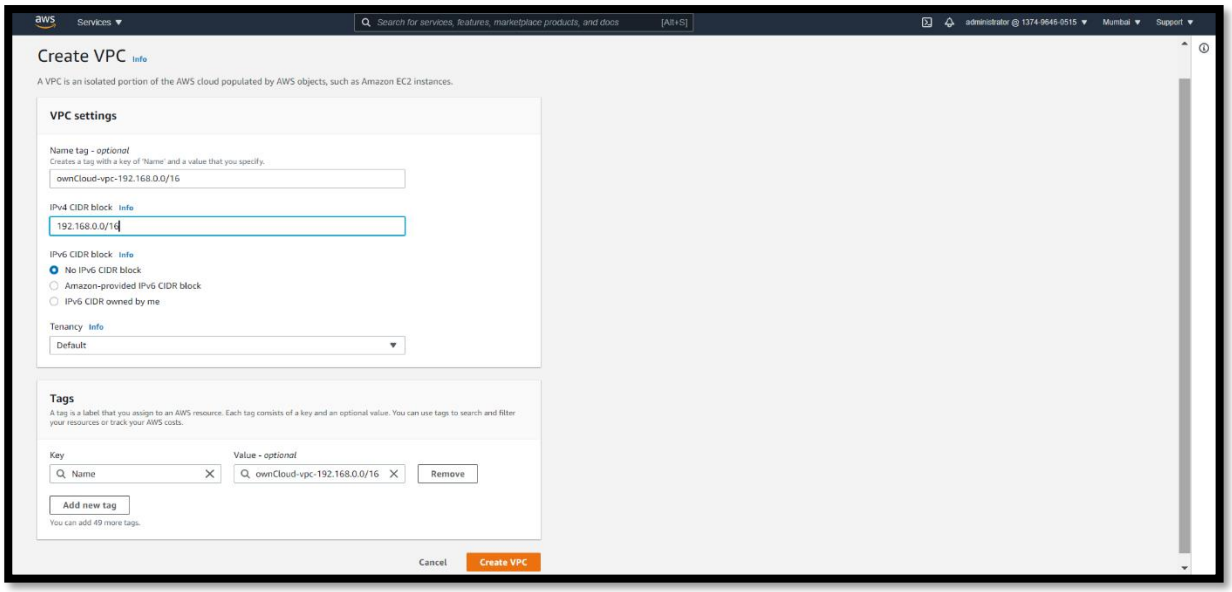
❖ Architecture:

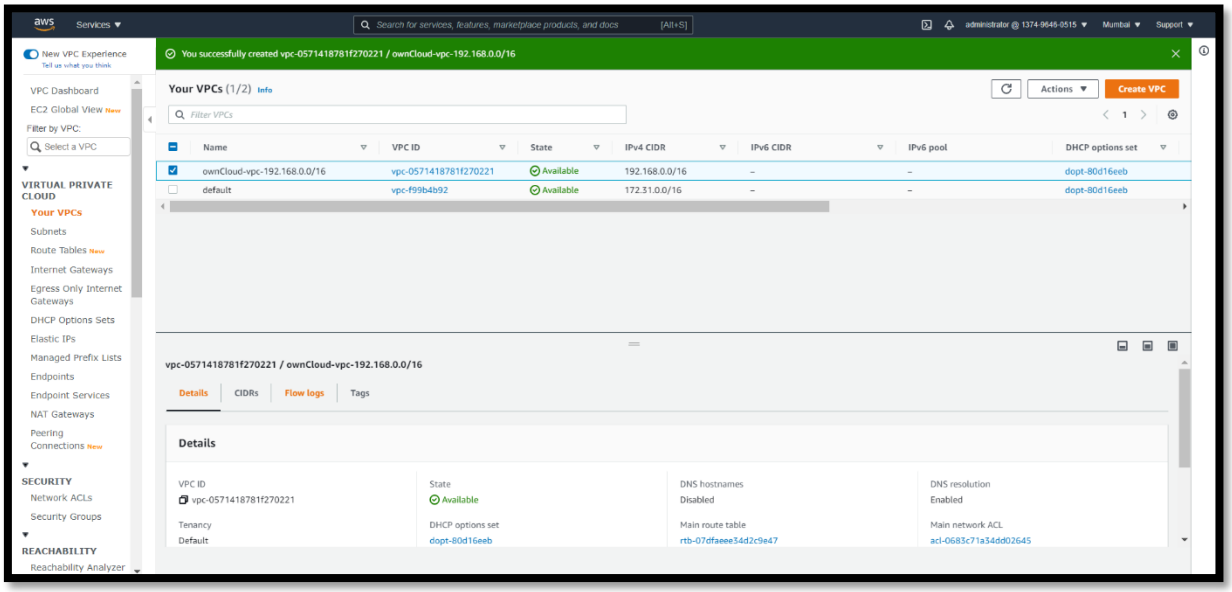


❖ VPC Creation:

- **Step1)** Login to AWC console.
- **Step2)** Type VPC on search bar.
- **Step3)** Create VPC with IPv4 CIDR as 192.168.0.0/16 and add tags for future purpose i.e., ***ownCloud-vpc-192.168.0.0/16***

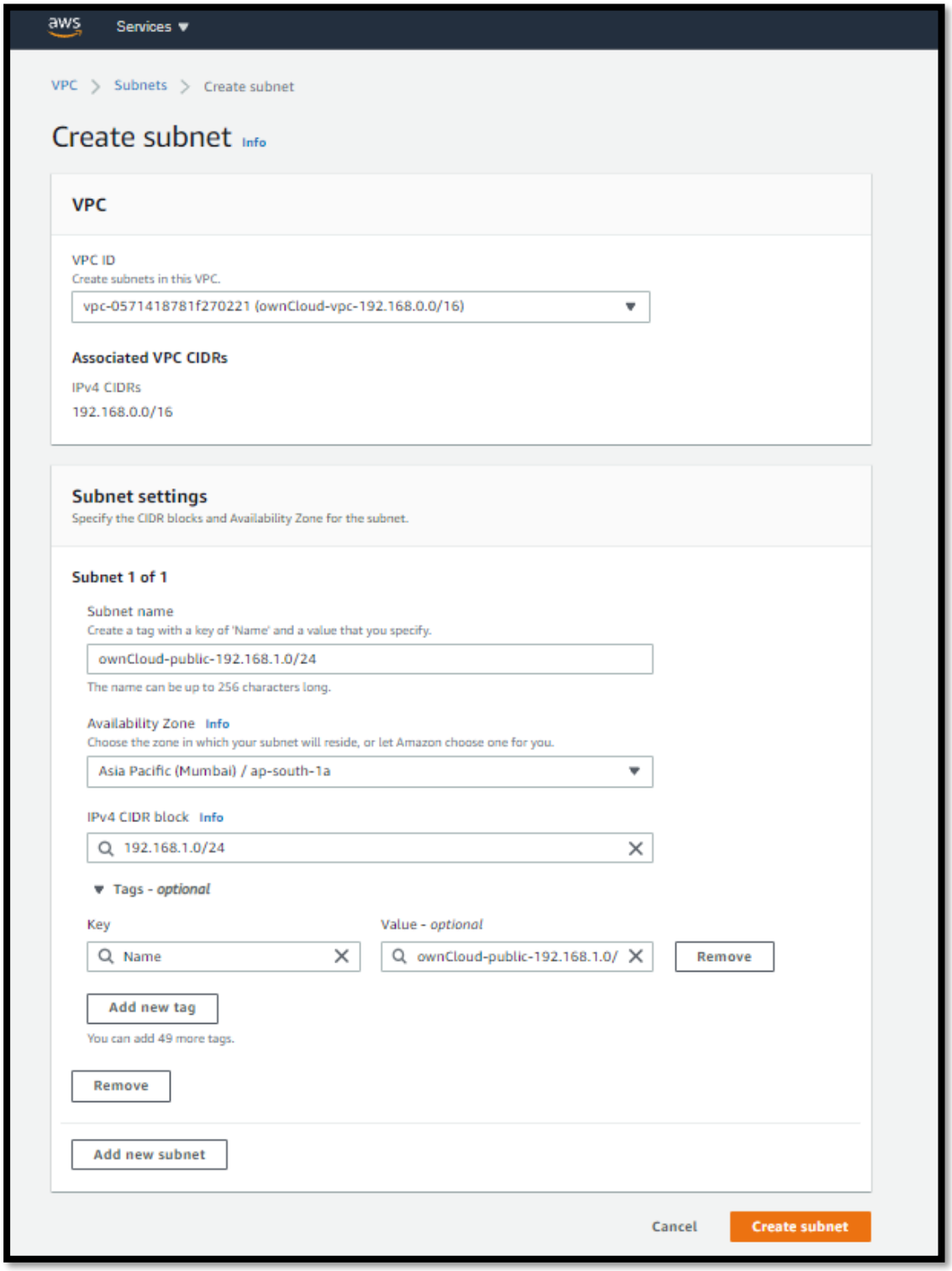




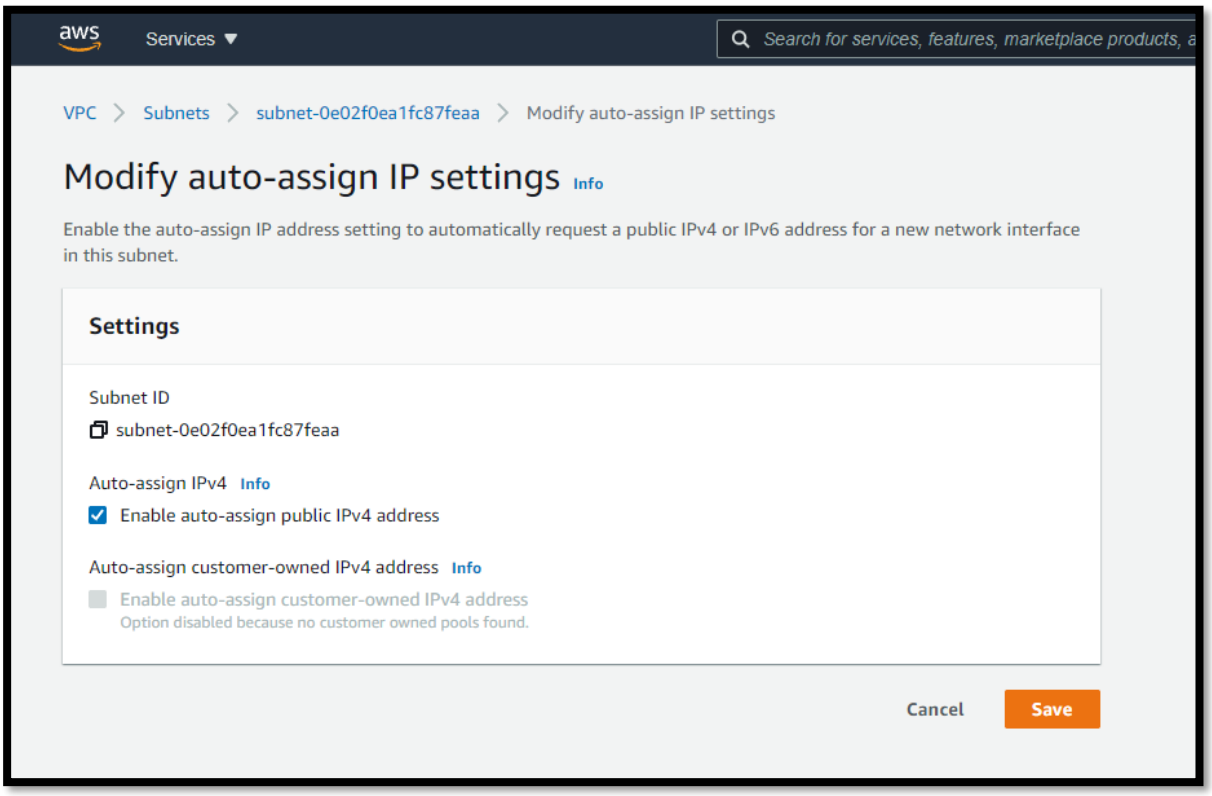


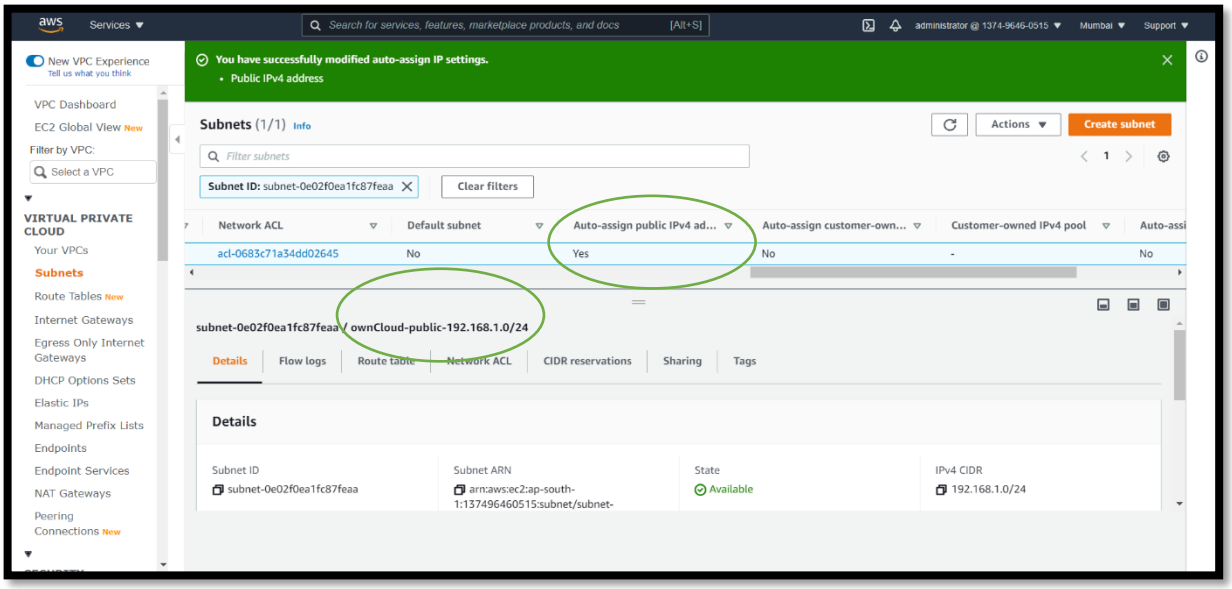
❖ Subnet Creation:

- **Step1)** Public subnet will have ***CIDR 192.168.1.0/24*** called ***ownCloud-public-192.168.1.0/24.*** Auto-assign IP address will be set as enabled.

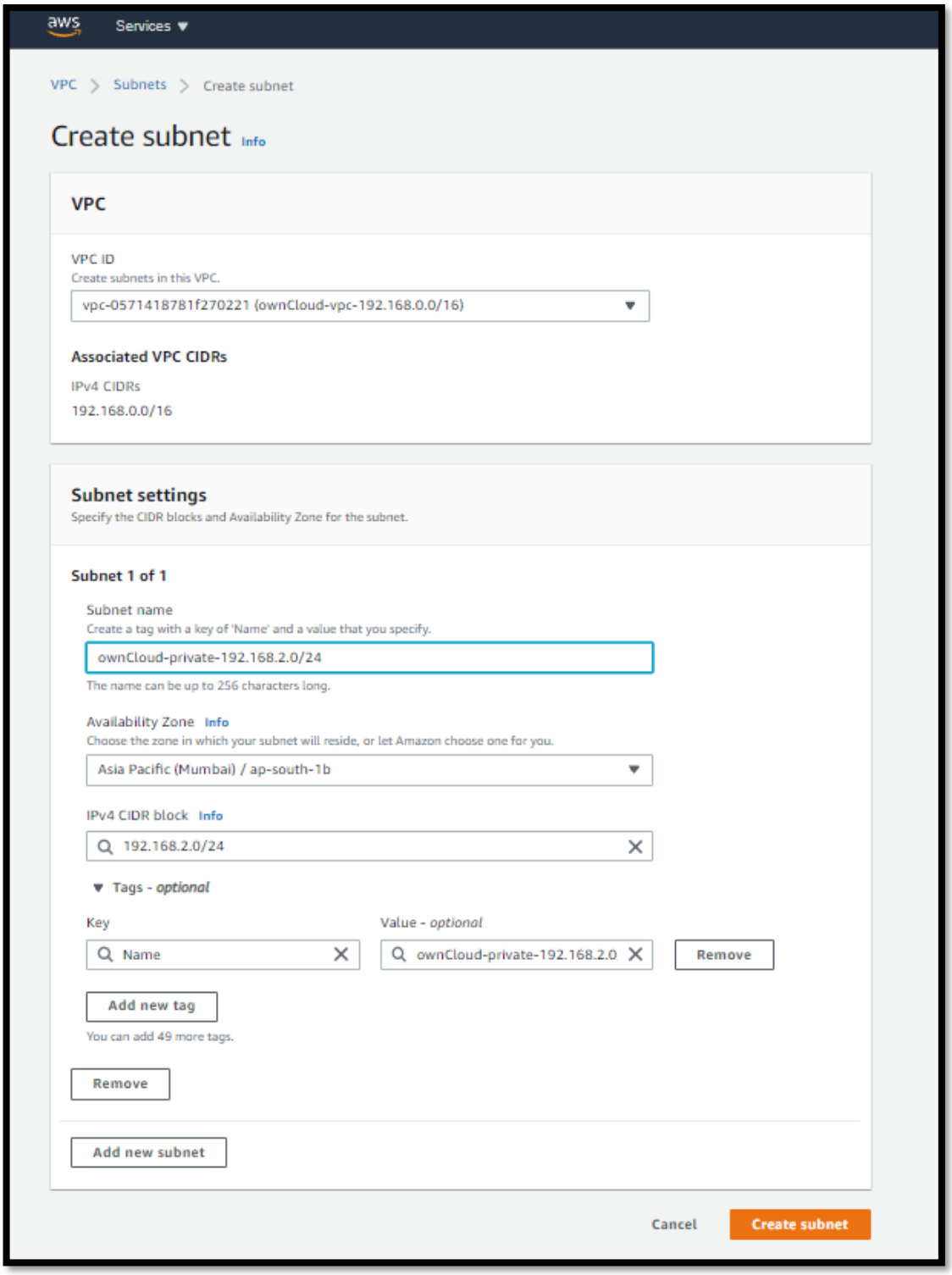


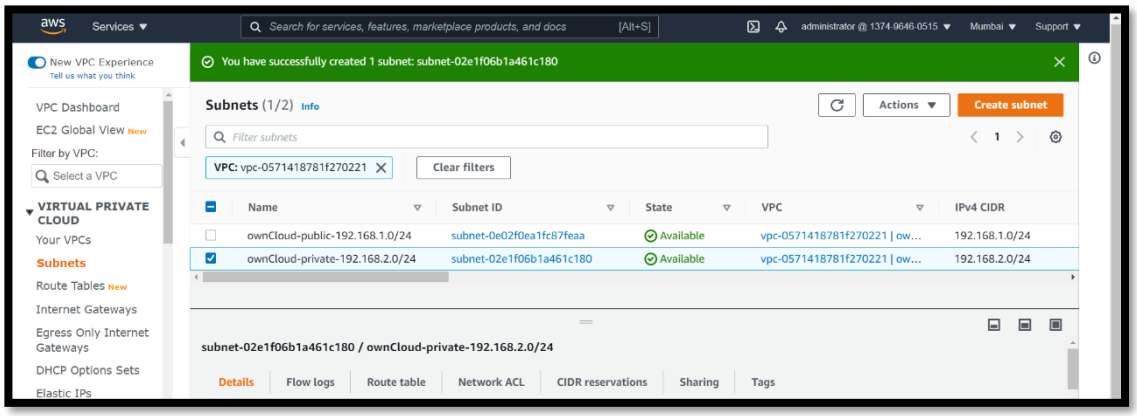
Auto-assign IP address will be set as enabled for Public Subnet.





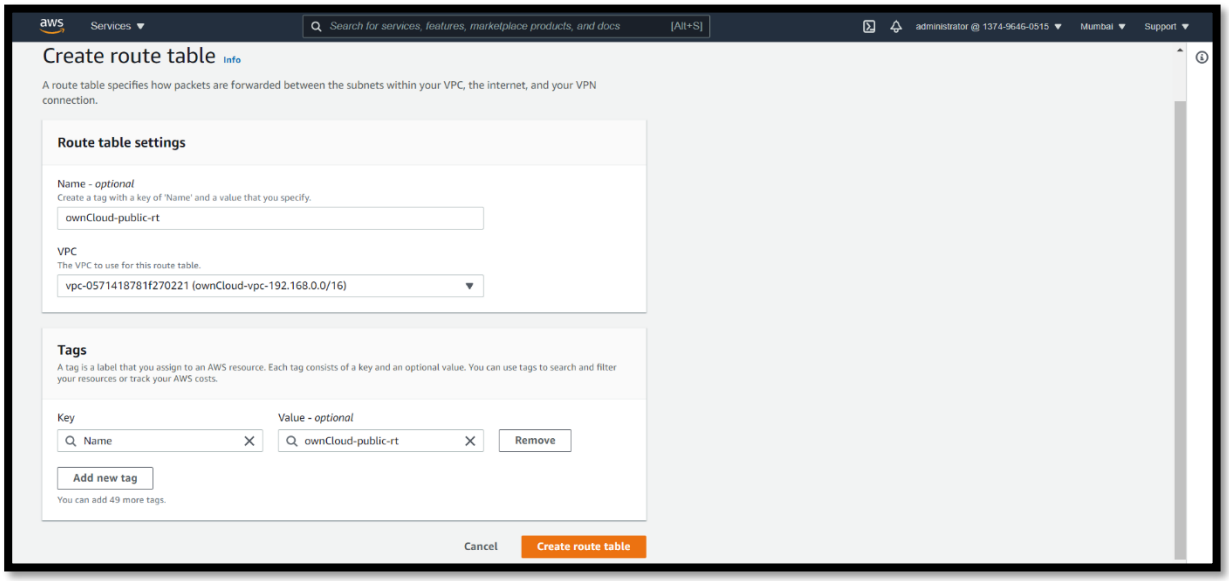
- **Step2)** Private subnet will have CIDR 192.168.2.0/24 called ***ownCloud-private-192.168.2.0/24.***

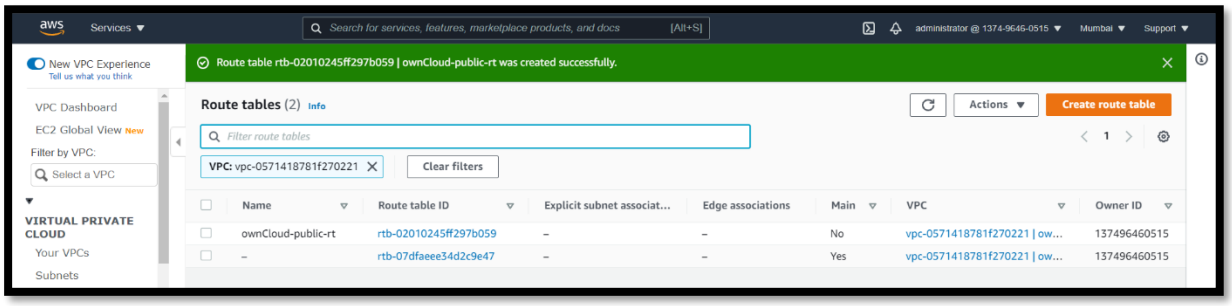

Services
VPC > Subnets > Create subnet
Create subnet Info
VPC
VPC ID
Create subnets in this VPC.
vpc-0571418781f270221 (ownCloud-vpc-192.168.0.0/16)
Associated VPC CIDRs
IPv4 CIDRs
192.168.0.0/16
Subnet settings
Specify the CIDR blocks and Availability Zone for the subnet.
Subnet 1 of 1
Subnet name
Create a tag with a key of 'Name' and a value that you specify.
ownCloud-private-192.168.2.0/24
The name can be up to 256 characters long.
Availability Zone Info
Choose the zone in which your subnet will reside, or let Amazon choose one for you.
Asia Pacific (Mumbai) / ap-south-1b
IPv4 CIDR block Info
192.168.2.0/24
Tags - optional
Key
Value - optional
Name
ownCloud-private-192.168.2.0
Remove
Add new tag
You can add 49 more tags.
Remove
Add new subnet
Cancel
Create subnet


Services
Search for services, features, marketplace products, and docs [Alt+S]
administrator @ 1374-9646-0515
Mumbai
Support
New VPC Experience
Tell us what you think
You have successfully created 1 subnet: subnet-02e1f06b1a461c180
VPC Dashboard
EC2 Global View New
Filter by VPC:
Select a VPC
VIRTUAL PRIVATE CLOUD
Your VPCs
Subnets
Route Tables New
Internet Gateways
Egress Only Internet Gateways
DHCP Options Sets
Elastic IPs
Subnets (1/2) Info
Actions
Create subnet
Filter subnets
VPC: vpc-0571418781f270221
Clear filters
Name
Subnet ID
State
VPC
IPv4 CIDR
ownCloud-public-192.168.1.0/24
subnet-0e02f0ea1fc87feaa
Available
vpc-0571418781f270221 | ow...
192.168.1.0/24
ownCloud-private-192.168.2.0/24
subnet-02e1f06b1a461c180
Available
vpc-0571418781f270221 | ow...
192.168.2.0/24
subnet-02e1f06b1a461c180 / ownCloud-private-192.168.2.0/24
Details
Flow logs
Route table
Network ACL
CIDR reservations
Sharing
Tags

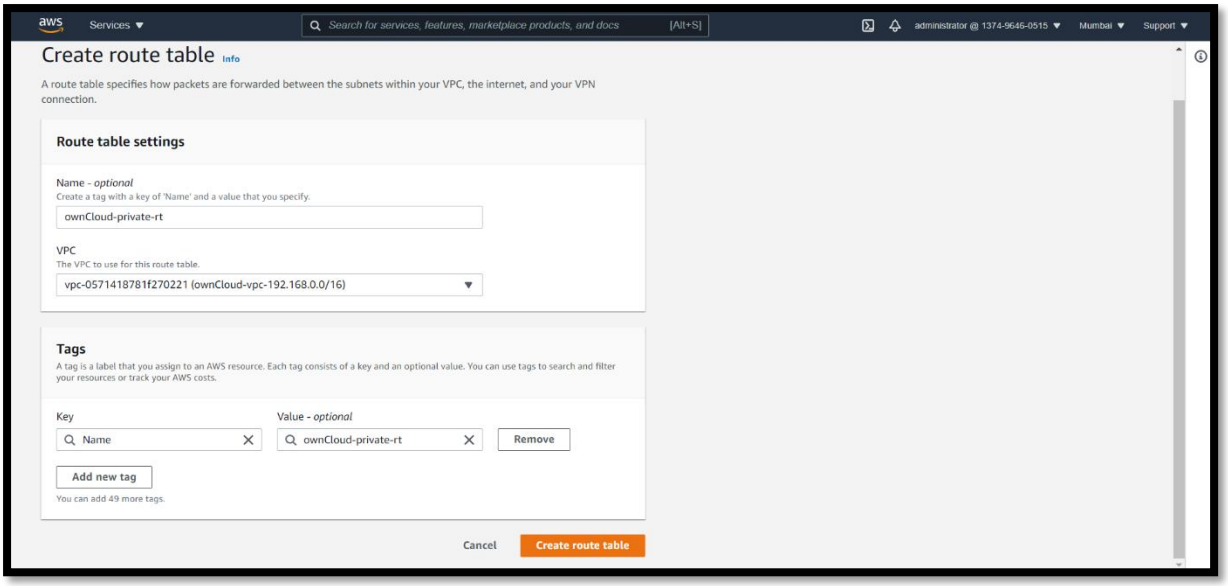
❖ Internet Gateway:

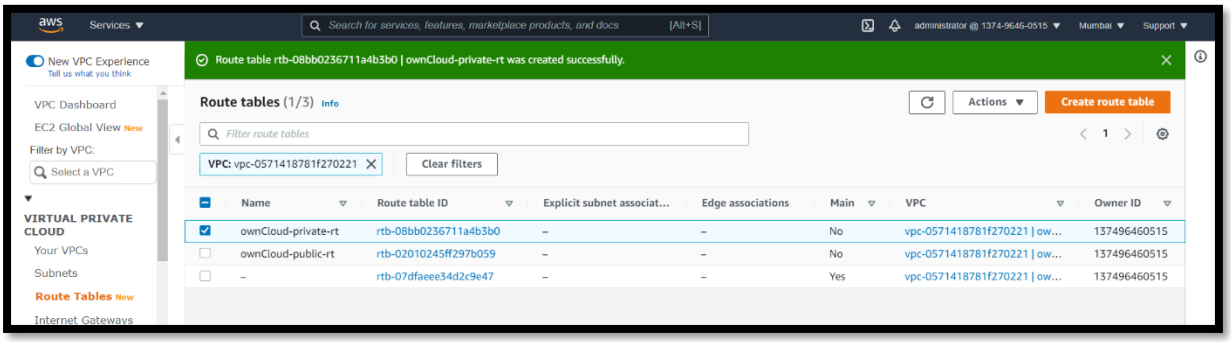
- **Step1)** Custom route table (***ownCloud-public-rt***) will be assigned to public subnet (***ownCloud-public-192.168.1.0/24***).



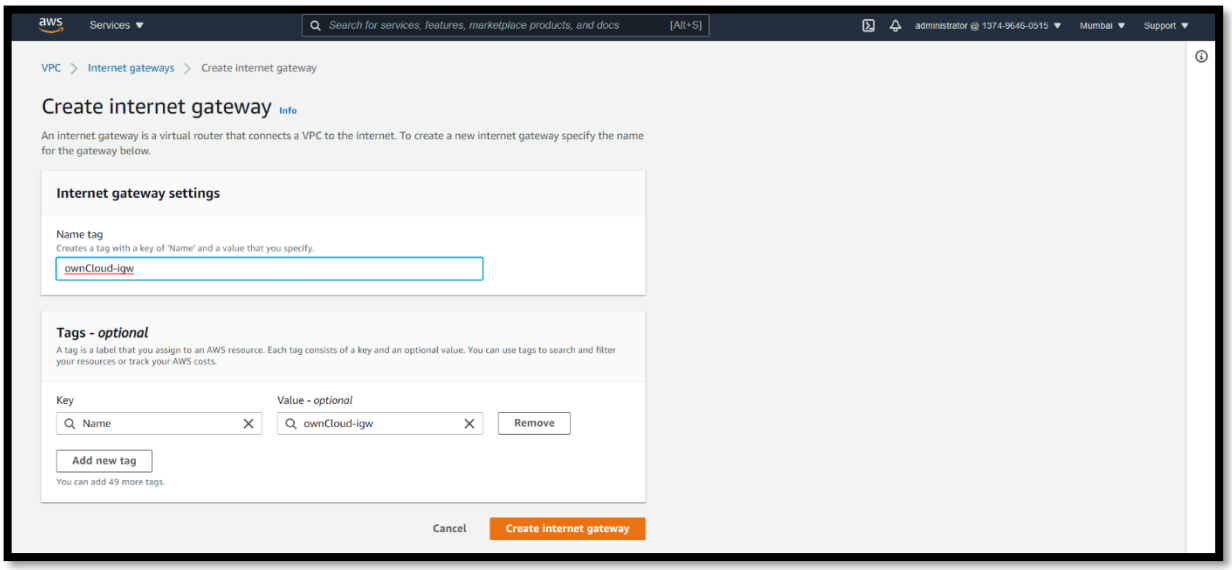


- **Step2)** Default route table (***ownCloud-private-rt***) will be assigned to Private subnet.

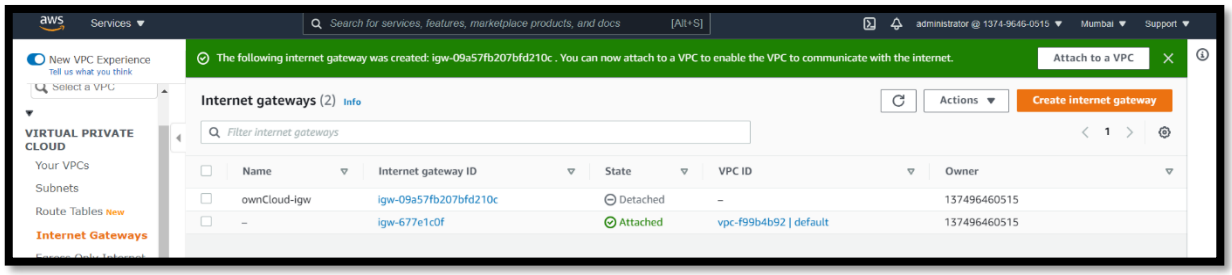


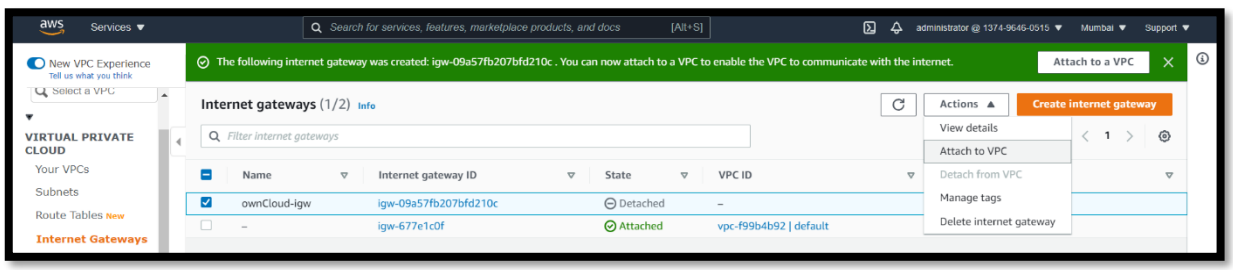


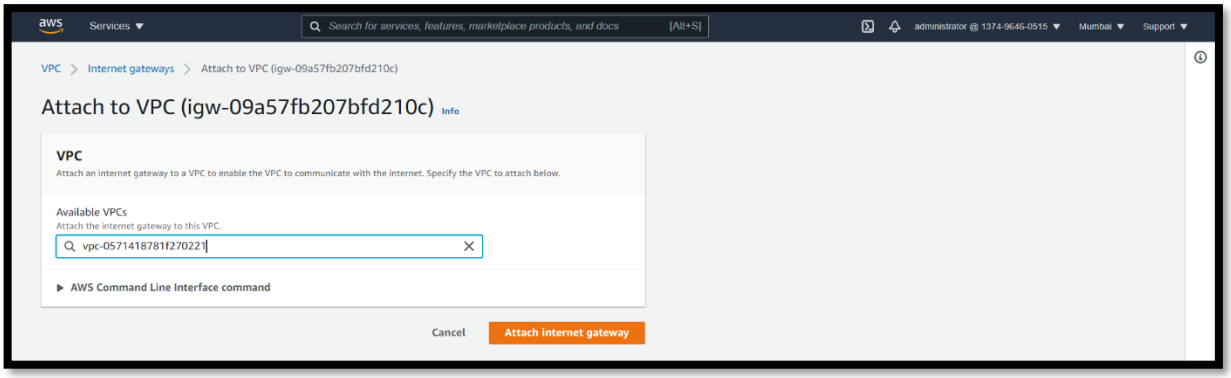
➢ **Step3)** Now we'll create internet-gateway (***ownCloud-igw***) and attach it to our VPC i.e., ***ownCloud-vpc-192.168.0.0/16***



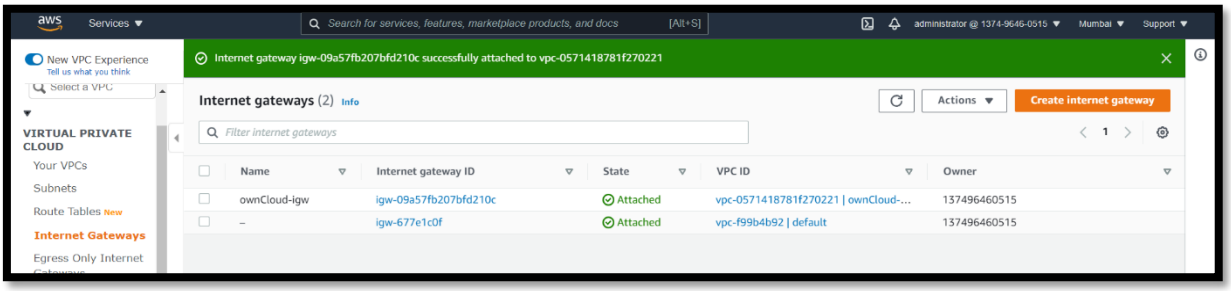
NOTE: The Gateway is in Detached state. We need to attach it to our VPC (***ownCloud-public-192.168.1.0/24*).**



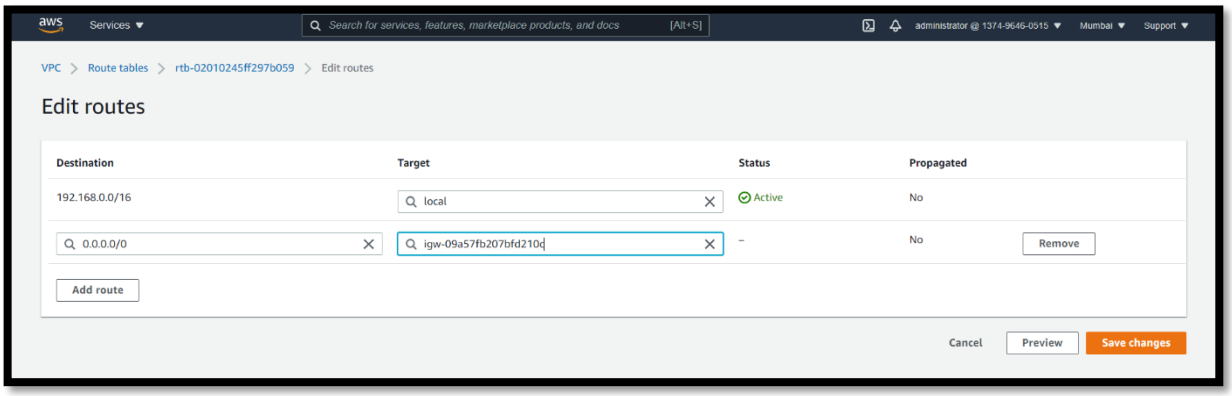


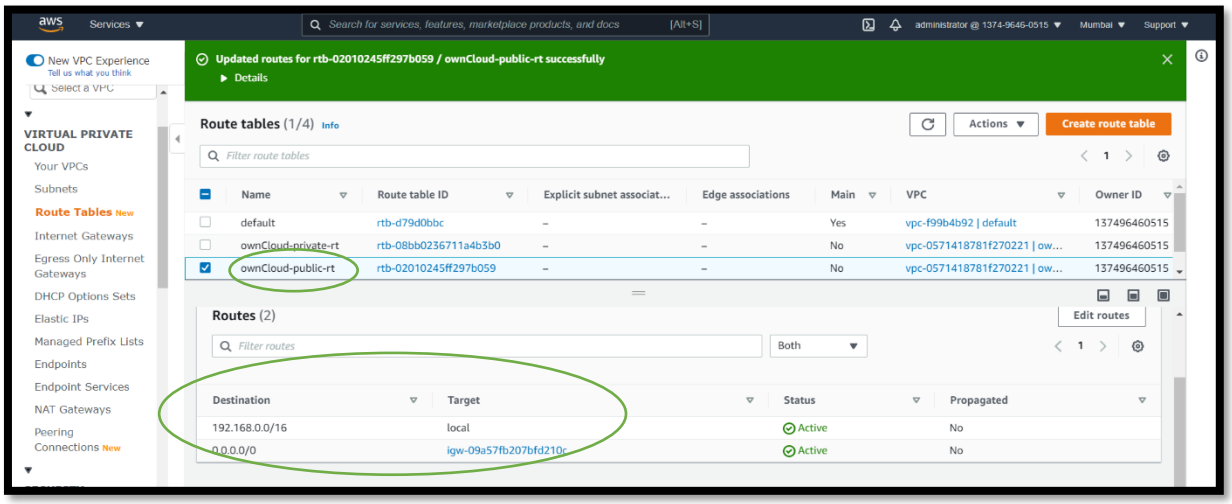


In the below image we can see that now our internet gateway is attached to our vpc.

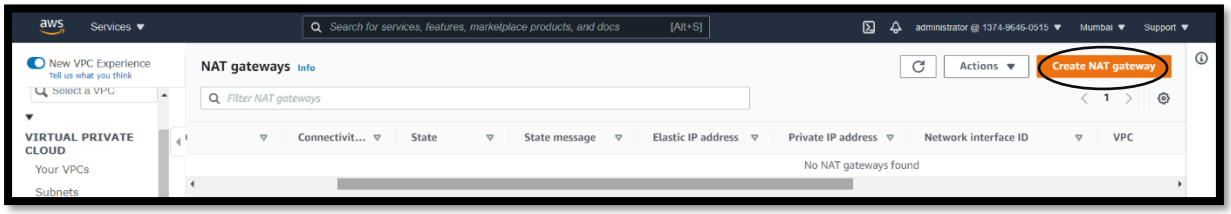


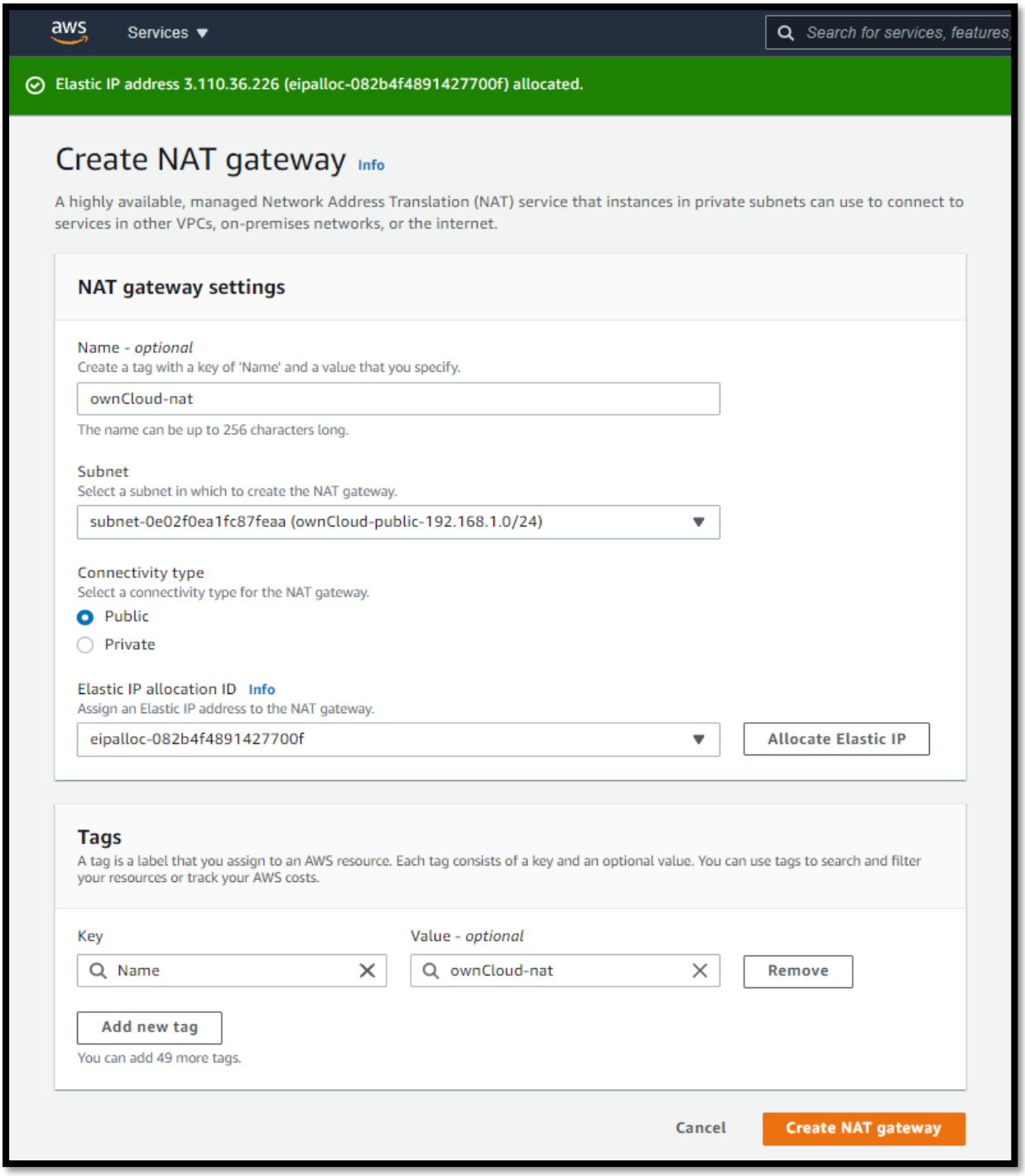
- **Step4)** ***ownCloud-public-rt*** will have bi-directional internet route entry using Amazon internet gateway. Also add pubic-subnet in ***Explicit Subnet Association***.

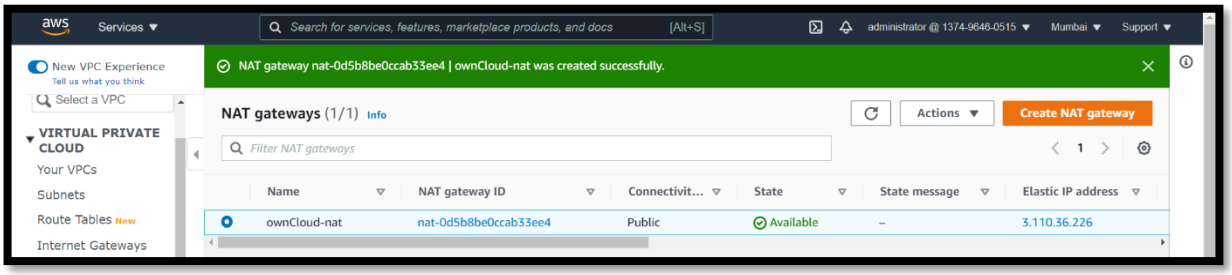




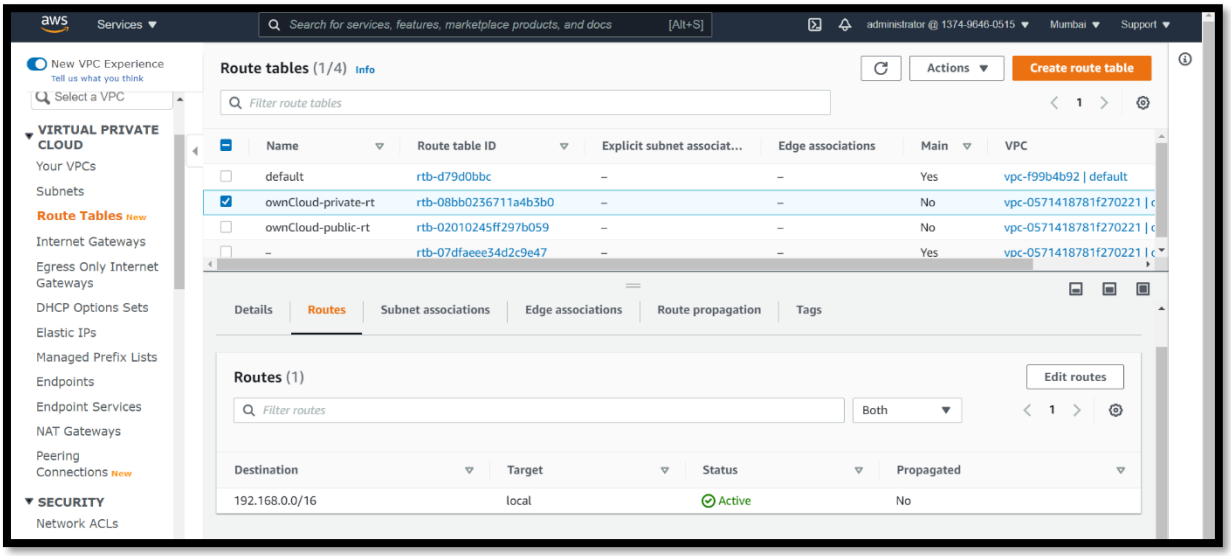
➢ **Step5)** Now I'll create a NAT gateway (***ownCloud-nat***) to facilitate internet connectivity to private subnet.

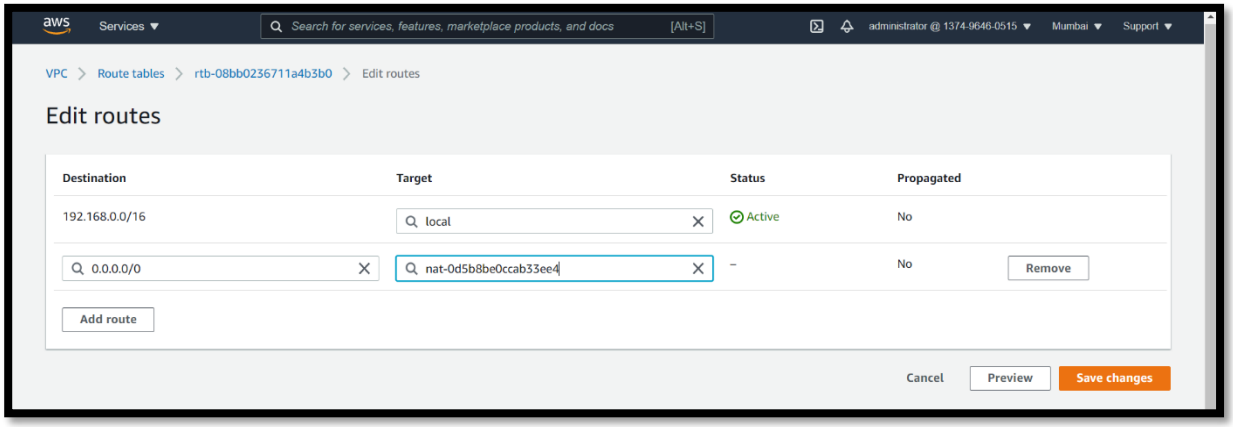


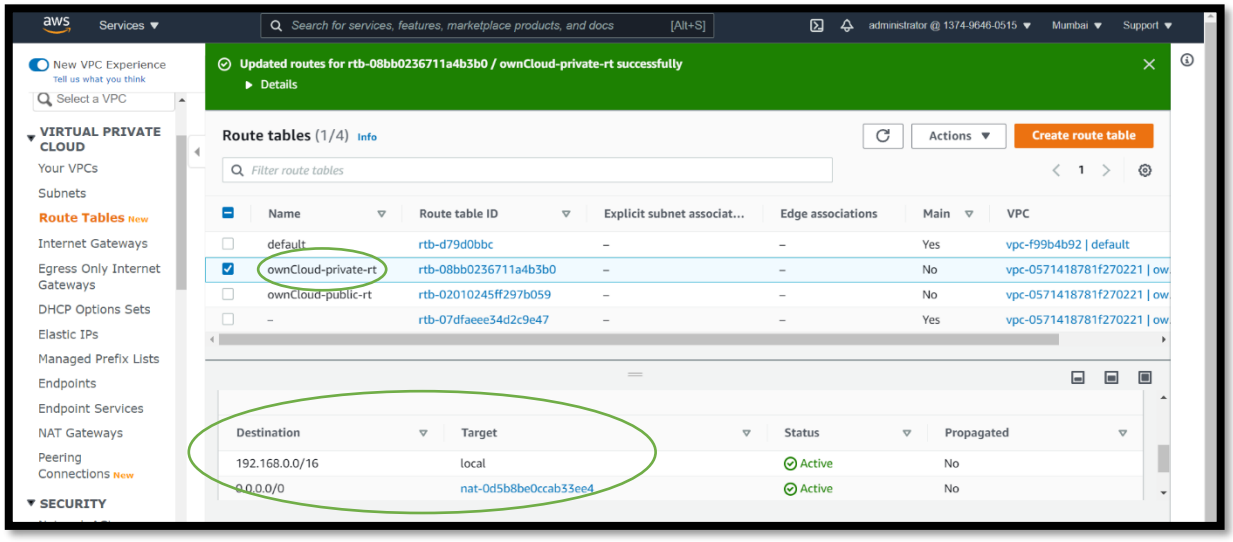




➢ **Step6)** ***ownCloud-private-rt*** will have NAT instance route entry to facilitate internet connectivity to private subnet.

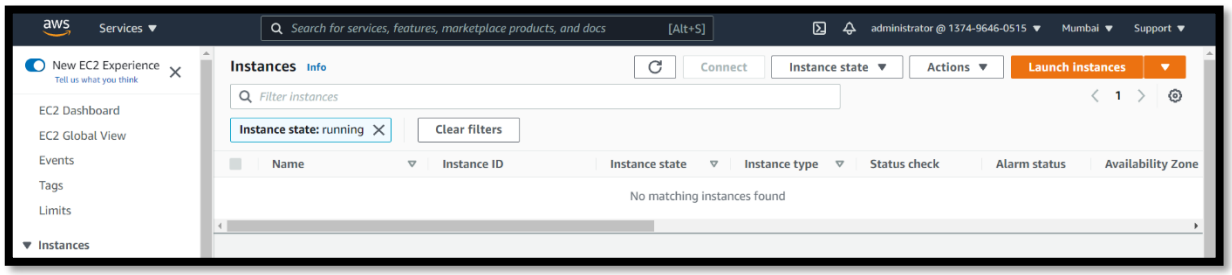




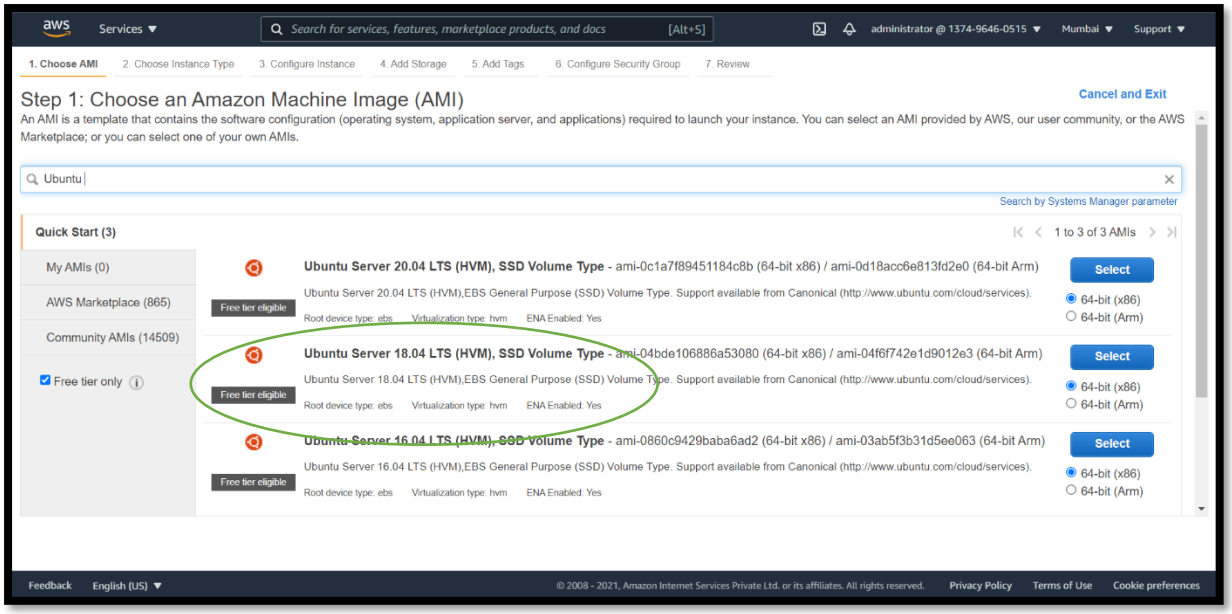


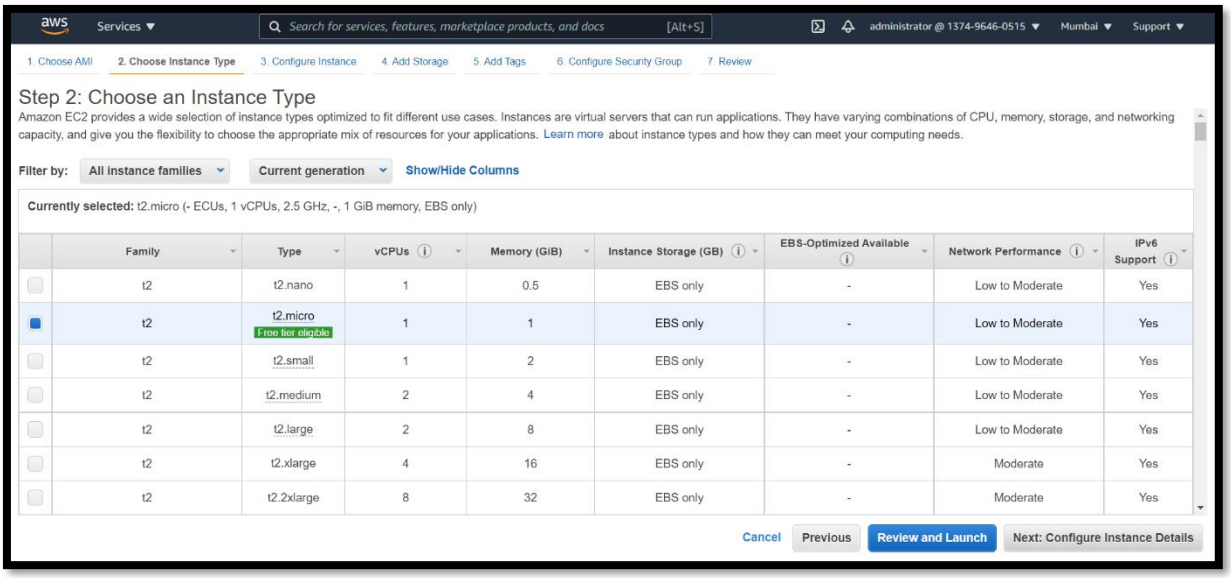
❖ Ec2 Instance Creation:

- **Step1)** I'll create 1st create one ***Ubuntu 18.04*** ec2-instance (***ownCloud-public-machine***) in public subnet.

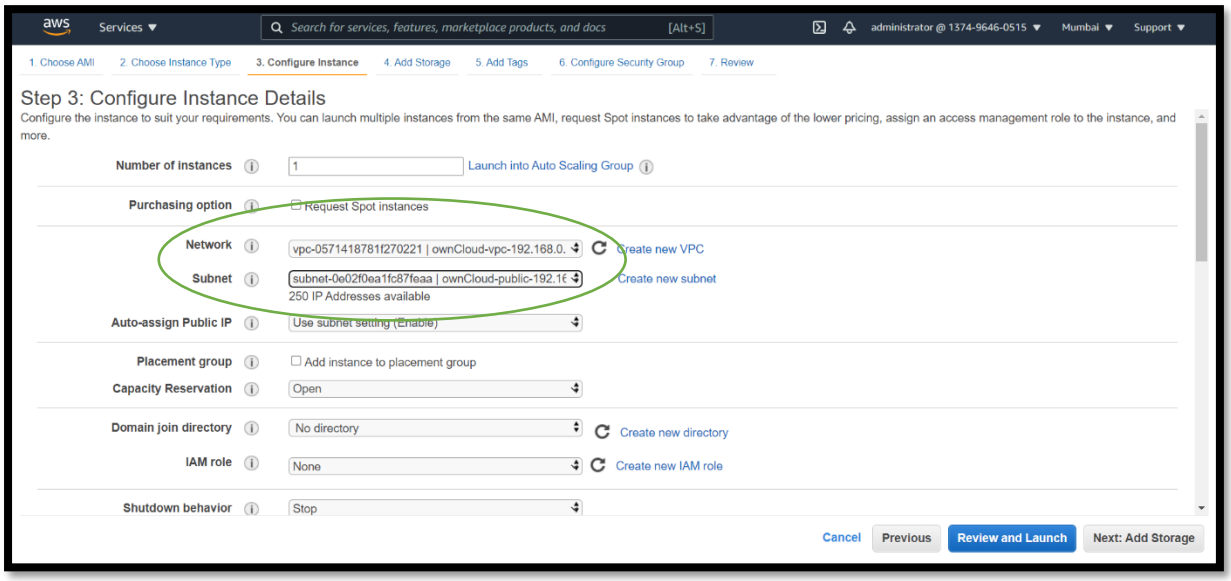


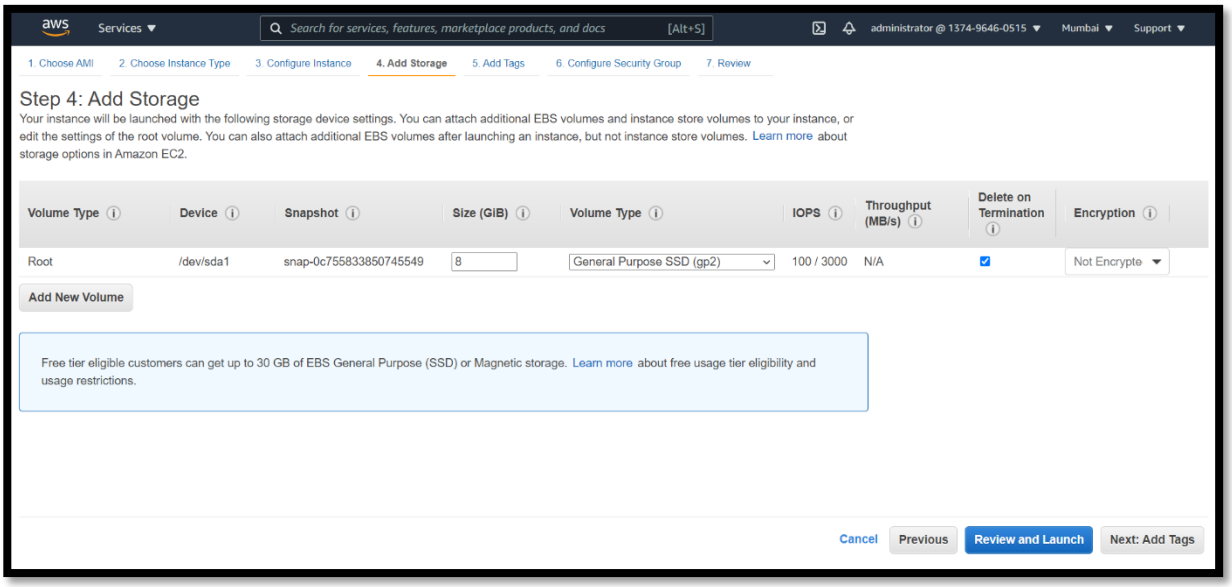
Selecting the Ubuntu 18.04 instance.

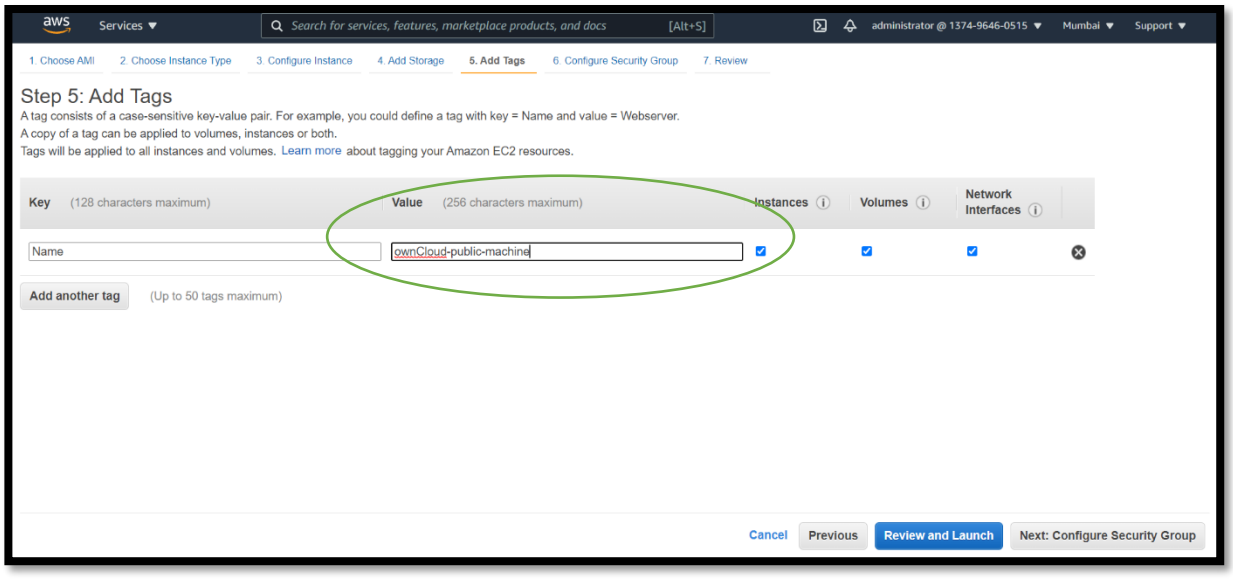




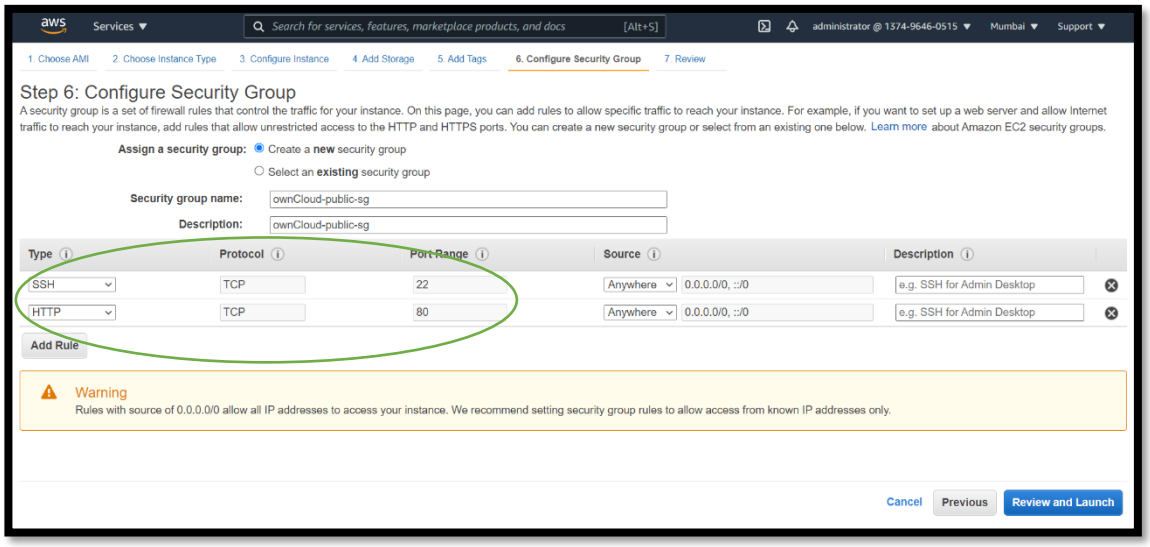
Selecting our ownCloud-vpc-192.168.0.0/16 in Network and Subnet as public subnet.



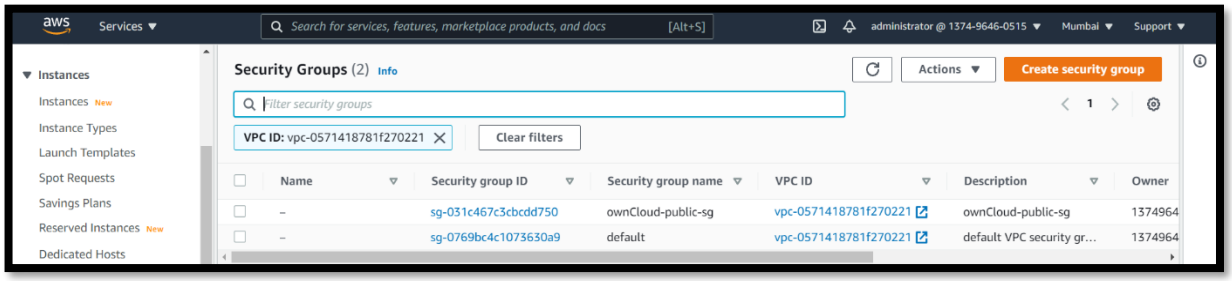


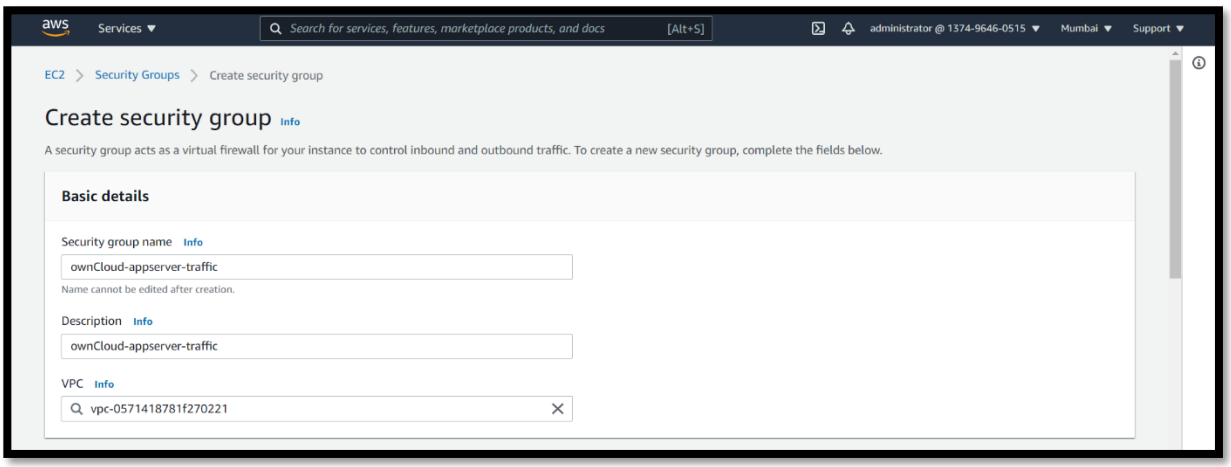


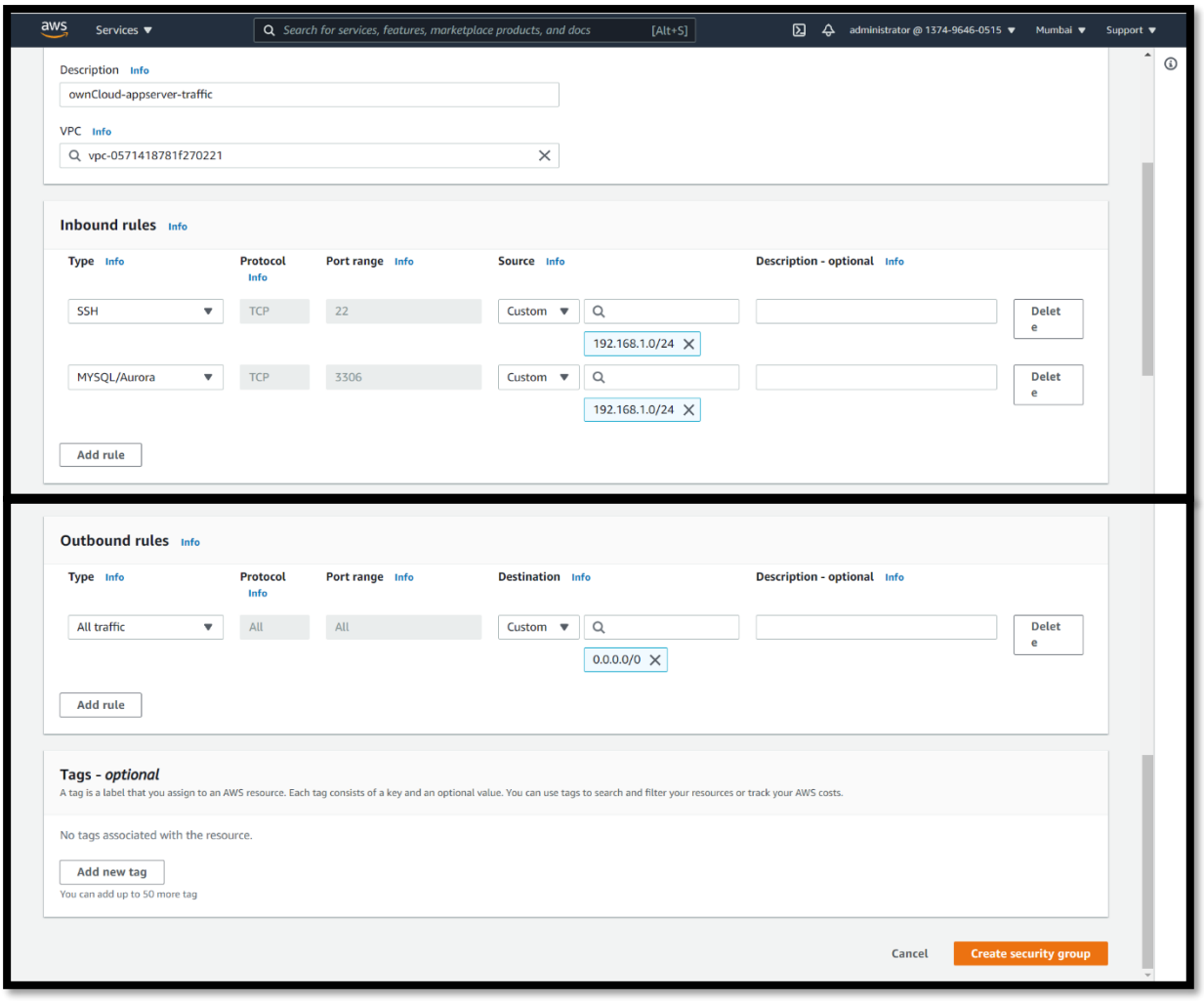
- **Step1.1)** Now I'll Add the Security group (***ownCloud-public-sg***) to the ownCloud-public-machine. This will be assigned to ownCloud web server EC2 instance in public subnet. It opens SSH port 22 for remote access and HTTP port 80 for web access. This opens unrestricted access to above ports for the world.



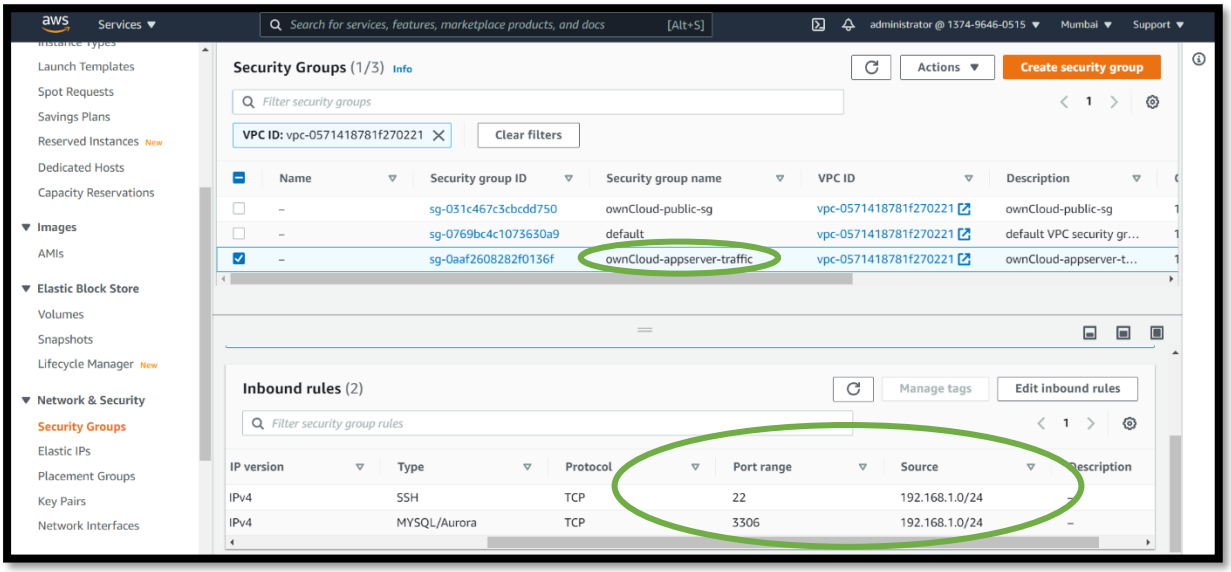
- **Step2)** Now I will create a security group (***ownCloud-appserver-traffic***) for my private machine This will be assigned to database server EC2 instance in private subnet. This will enable restricted access to server from public subnet only. It opens SSH port 22 for remote access and MYSQL DB port 3306 for remote database connection.



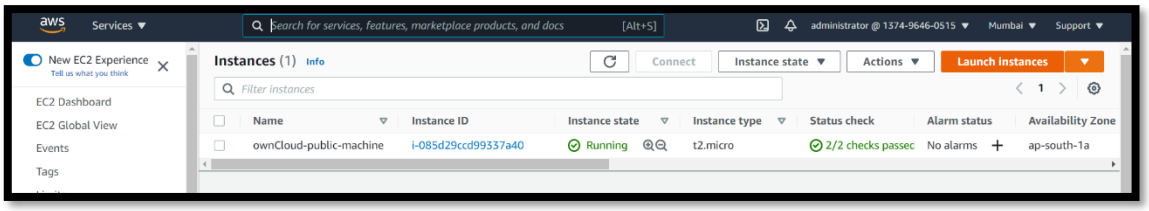




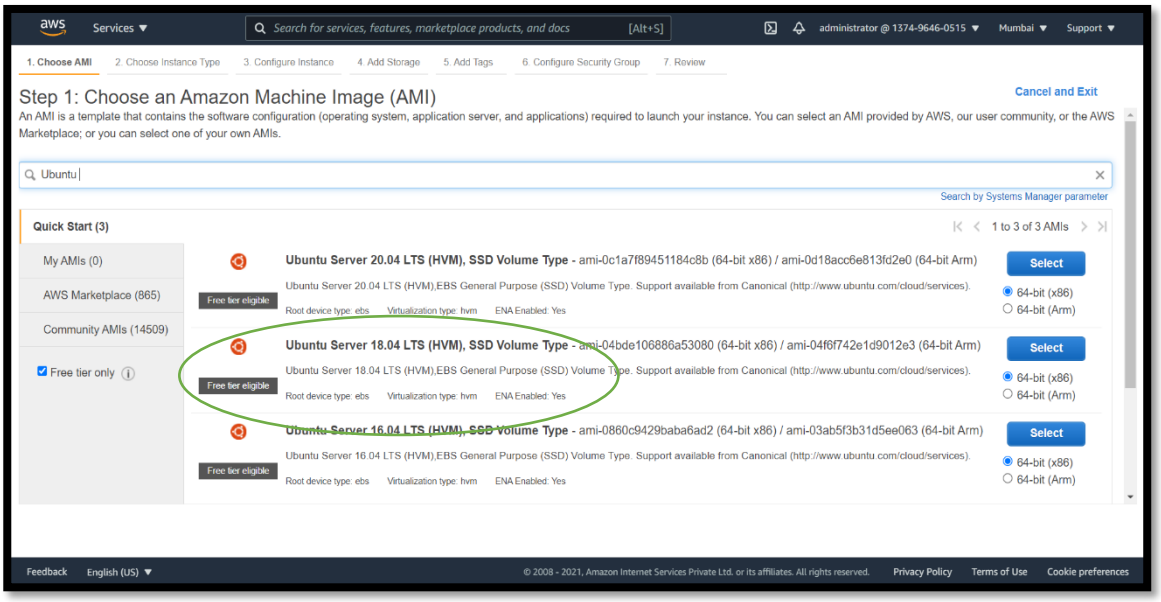
Note: We can see in the below image that security group ***ownCloud-appserver-traffic*** machine could only be accessed by our public instance machine.

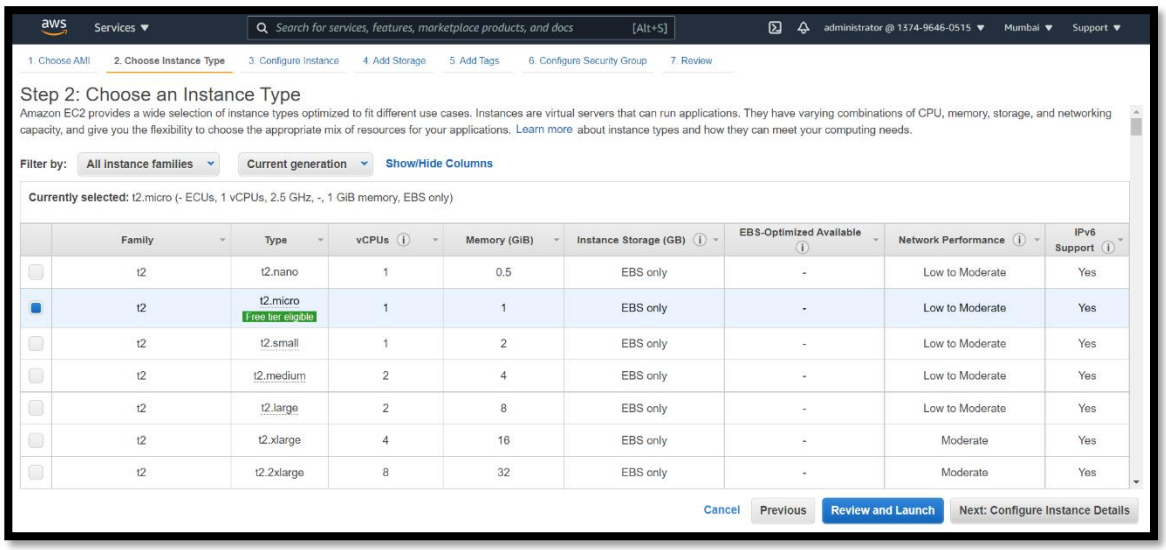


- **Step2.1)** Now I'll Create my ec2-machine in private subnet with name ***ownCloud-privateDB-machine*** and give it the security group which I created in step 2. i.e., ownCloud-appserver-traffic.

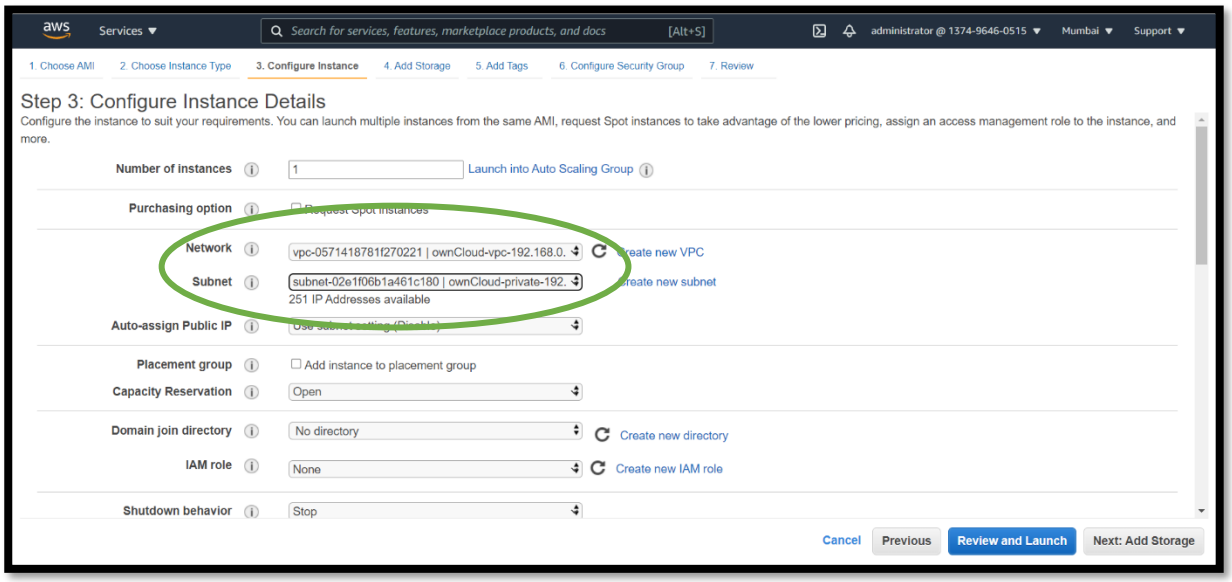


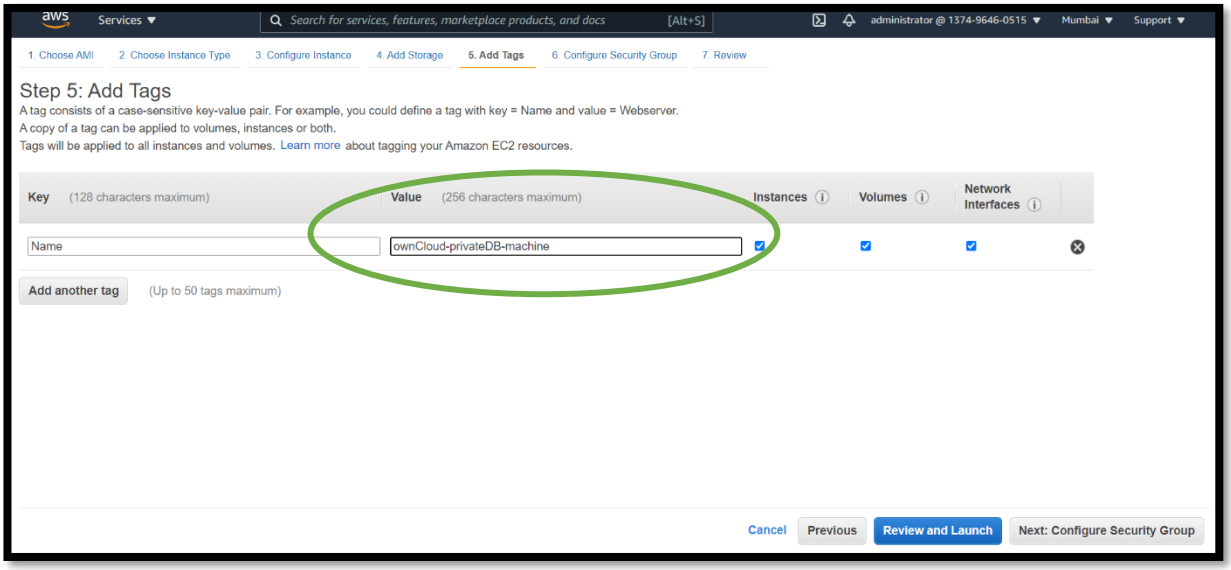
Selecting the Ubuntu 18.04 instance.



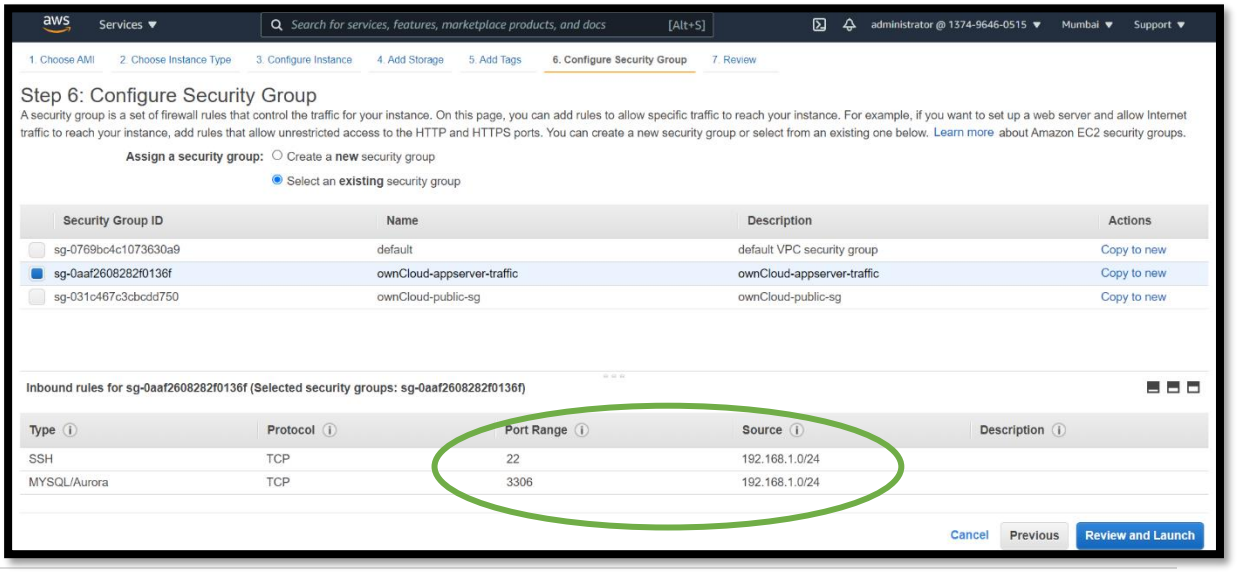


Selecting our ownCloud-vpc-192.168.0.0/16 in Network and Subnet as public subnet.



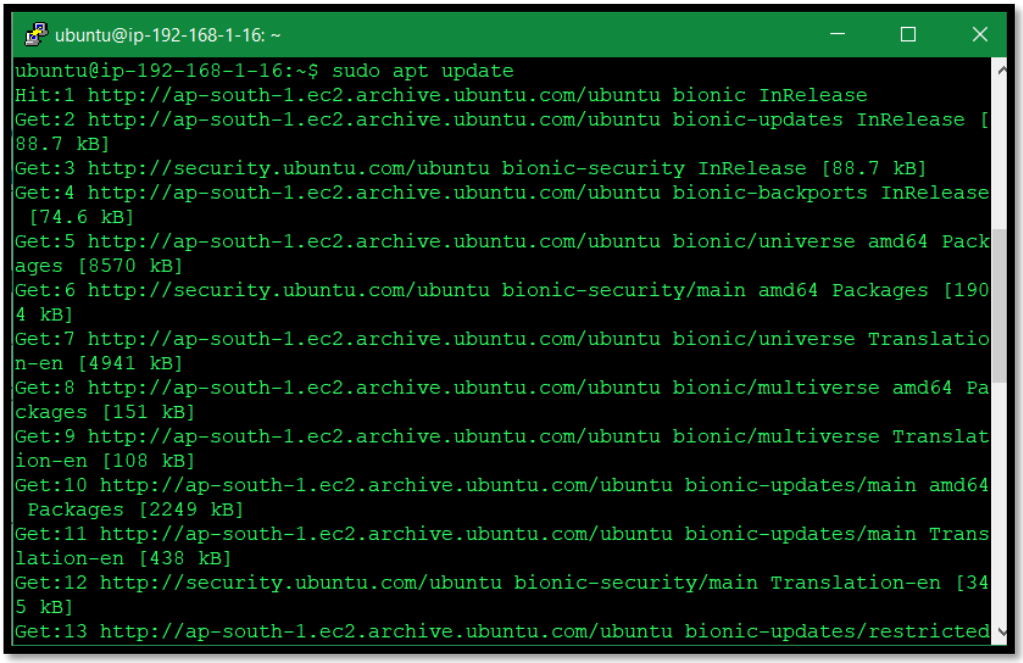


We can see below that restricted access is allowed to server from public subnet only. It opens
. SSH port 22 for remote access and MYSQL DB port 3306 for remote database connection.

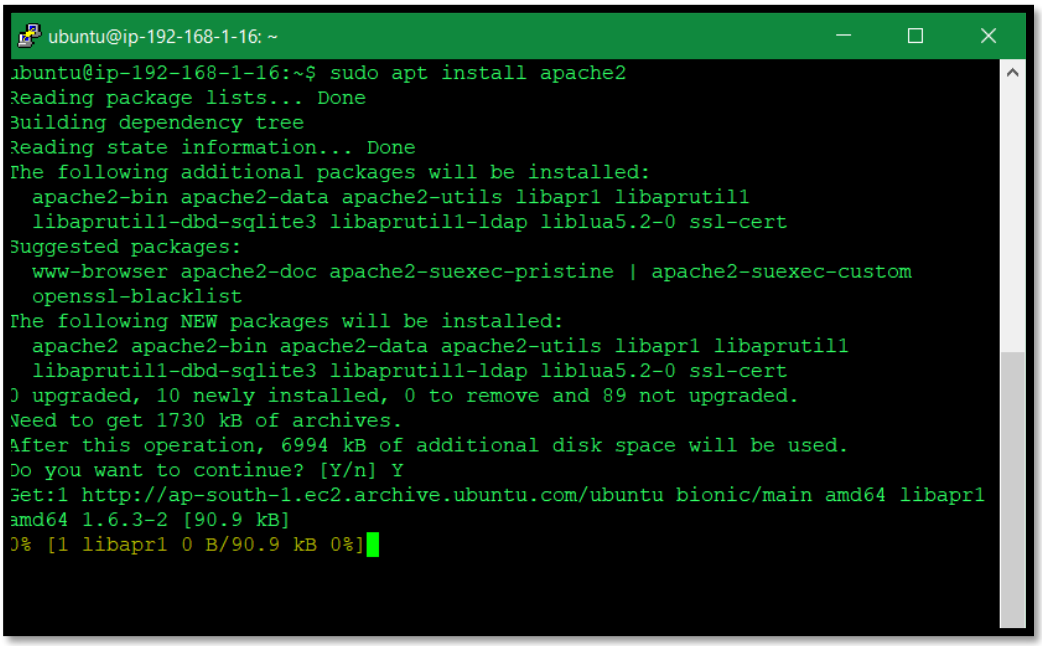


❖ Installing Apache and Updating Firewall (if required):

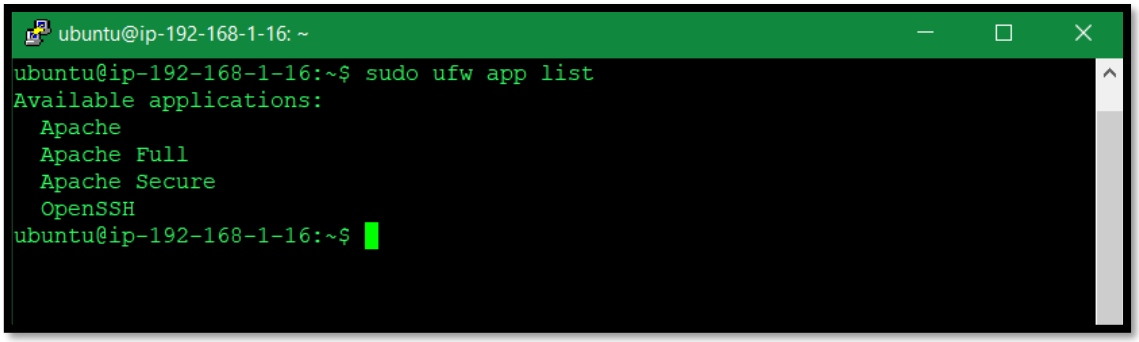
- **Step1)** First make sure your apt cache is updated with command: ***sudo apt update.***



- **Step2)** Now I will install Apache Web server with command: **sudo apt install apache2**



➢ **Step3)** Next, I'll check that UFW firewall is enabled or not, to make sure that our firewall allows HTTP and HTTPS traffic. We can check with the command: ***sudo ufw app list***



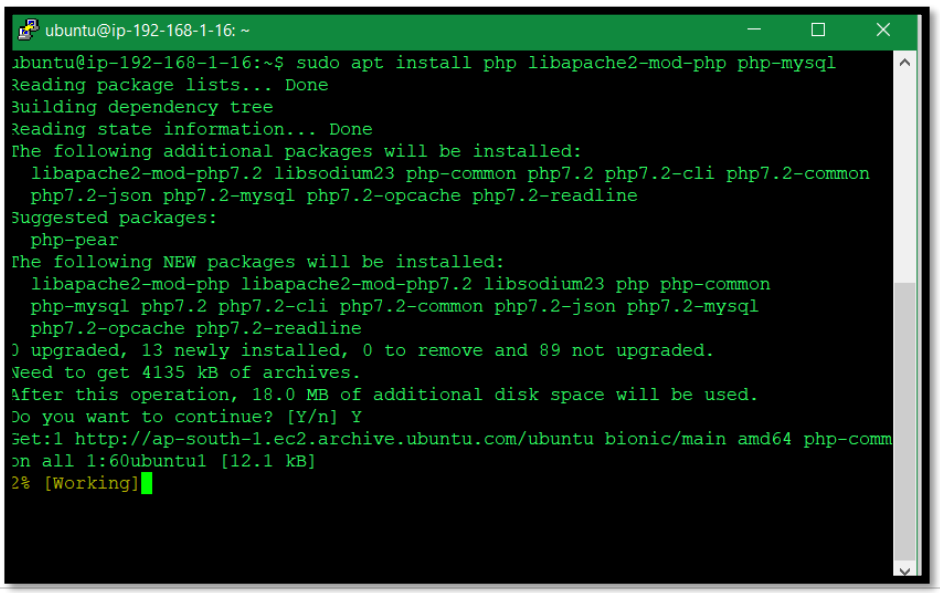
In above output Apache Full is there that means ports 80 and 443 are enabled.

❖ Installing Php:

PHP is the component of your setup that will process code to display dynamic content. It can run scripts, connect to your MySQL databases to get information, and hand the processed content over to your web server so that it can display the results to your visitors.

Once again, leverage the apt system to install PHP. In addition to the php package, we'll also need libapache2-mod-php to integrate PHP into Apache, and the php-mysql package to allow PHP to connect to MySQL databases. Run the following command to install all three packages and their dependencies:

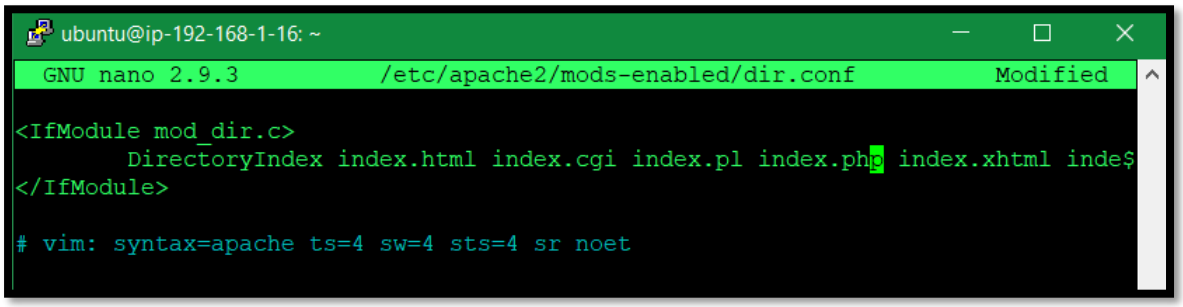
➢ **Step1)** To install PHP use the command: ***sudo apt install php libapache2-mod-php php-mysql***



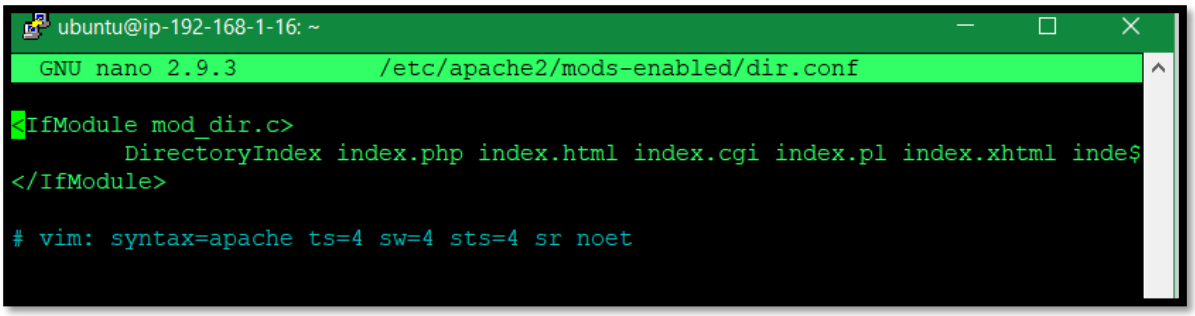
❖ Changing Apache's Directory Index:

In some cases, you'll want to modify the way that Apache serves files when a directory is requested. Currently, if a user requests a directory from the server, Apache will first look for a file called index.html. We want to tell the web server to prefer PHP files over others, to make Apache look for an index.php file first. If you don't do that, an index.html file placed in the document root of the application will always take precedence over an index.php file. To make this change, open the dir.conf configuration file in a text editor.

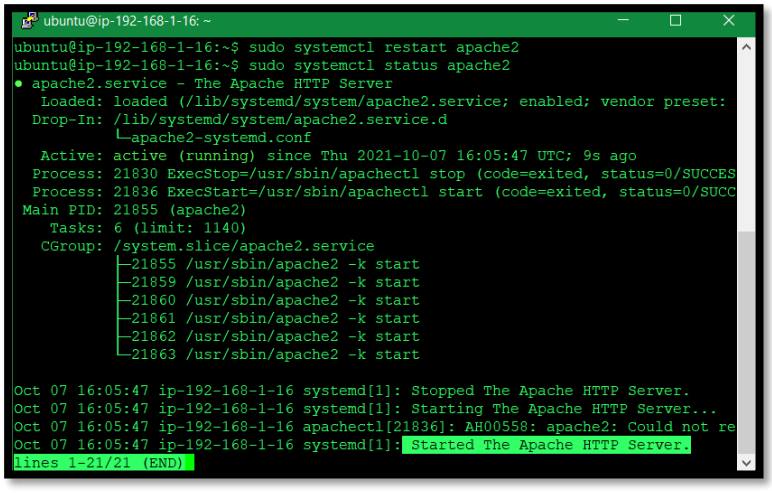
Before:



After:



- **Step1)** After this, restart the Apache web server in order for our changes to be recognized and check its status

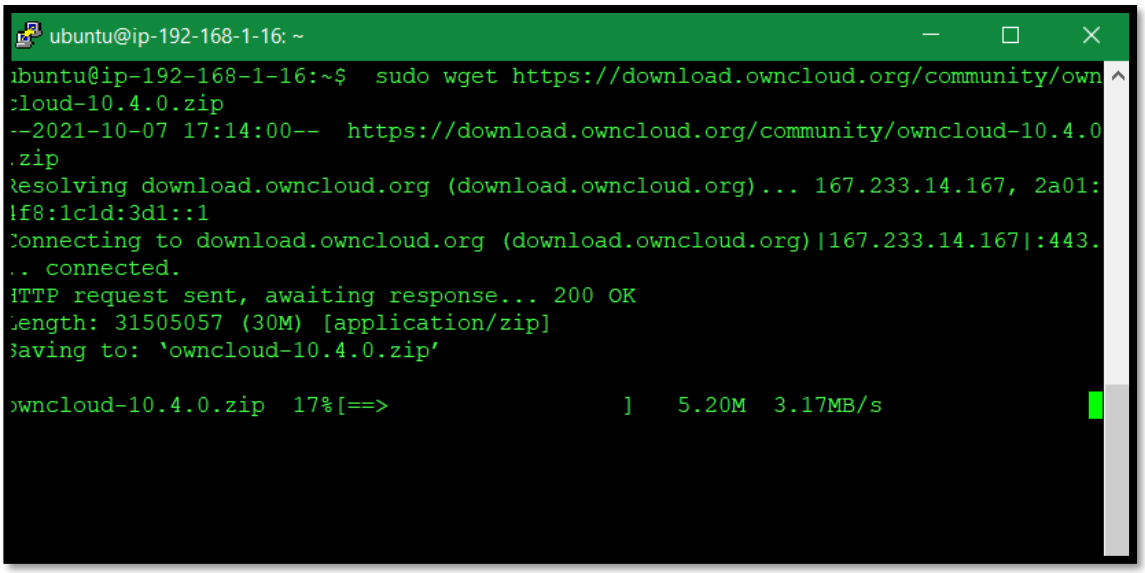


❖ Installing ownCloud in Public Machine:

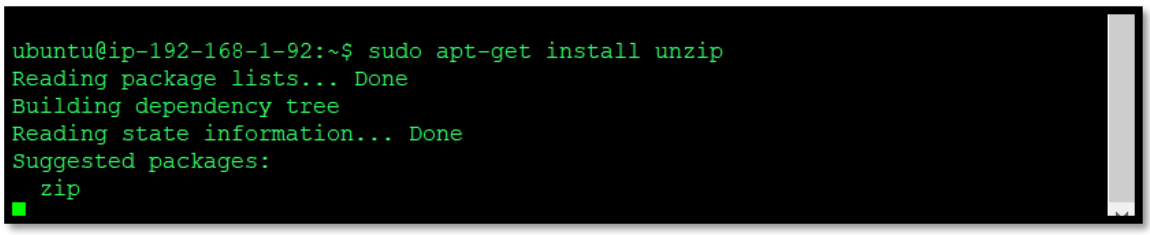
The ownCloud server package does not exist within the default repositories for Ubuntu. However, ownCloud maintains a dedicated repository for the distribution that we can add to our server.

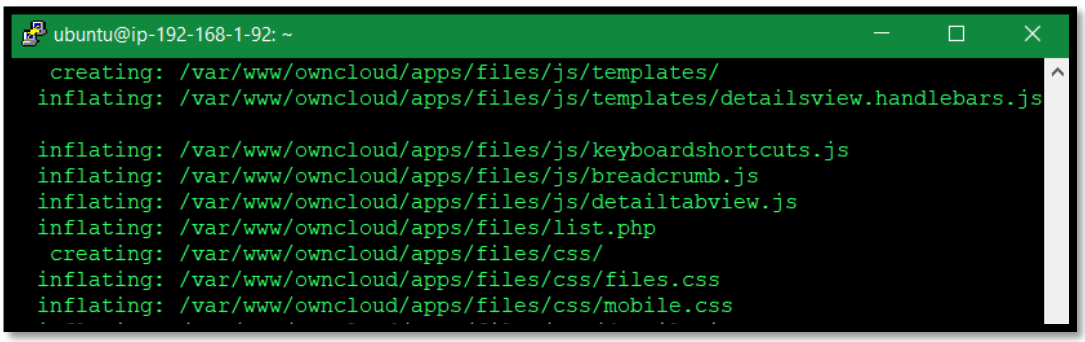
To begin, download their release key using the curl command and import it with the apt-key utility with the add command:

➢ **Step1)** Downloading zip file of ownCloud on Ubuntu machine.



➢ **Step2)** Once downloaded, unzip the zipped package to the /var/www/ directory, but 1st we need to install unzip package.

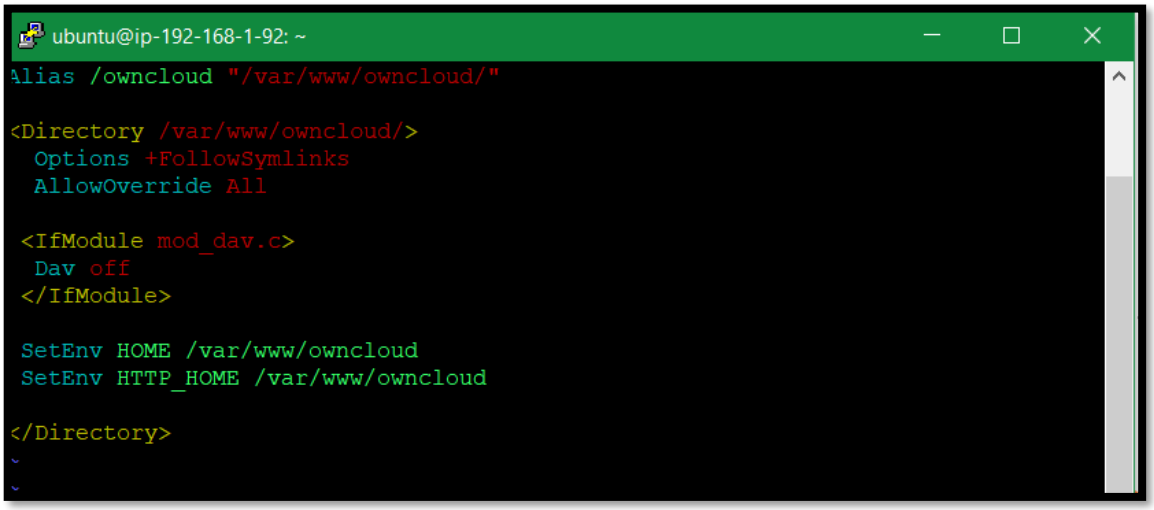




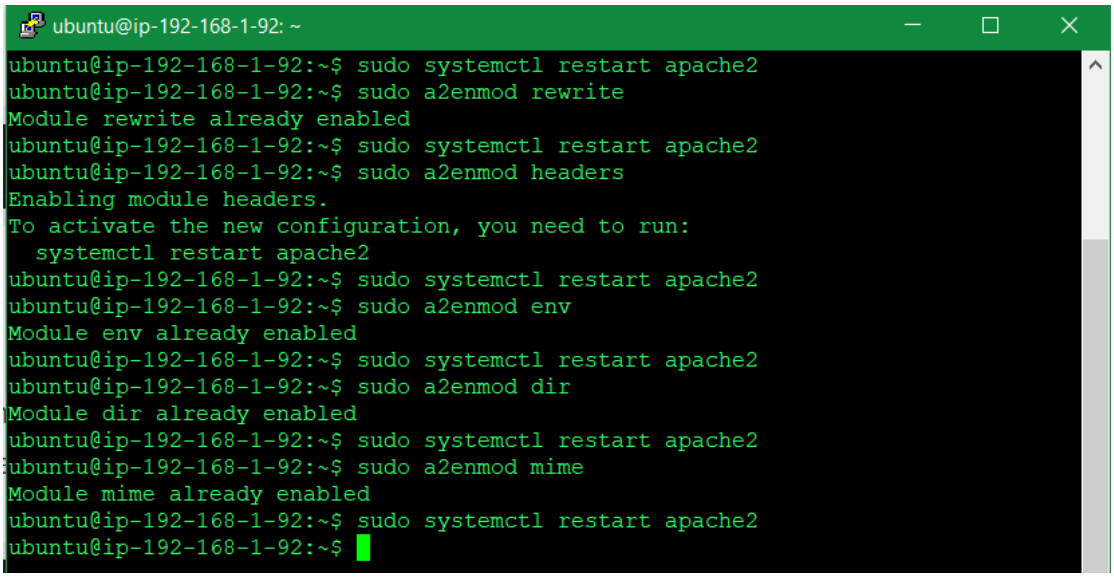
❖ Configuring Apache for ownCloud:

➢ **Step1)** Creating a configuration file (owncloud.cnf) for owncloud and adding the configuration given below:

```
Alias /owncloud "/var/www/owncloud/"

<Directory /var/www/owncloud/>
 Options +FollowSymlinks
 AllowOverride All

 <IfModule mod_dav.c>
 Dav off
 </IfModule>

 SetEnv HOME /var/www/owncloud
 SetEnv HTTP_HOME /var/www/owncloud

</Directory>
```



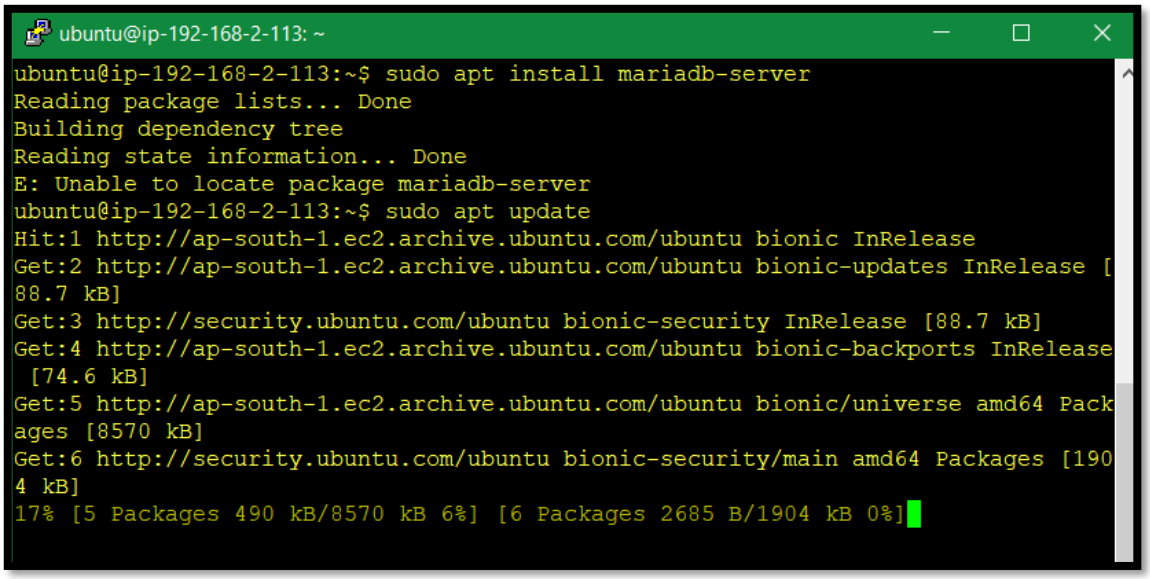
➢ **Step2)** Enabling all the required Apache modules and the newly added configuration & For the changes to come into effect restart the Apache webserver.



❖ Installing MySQL/MariaDB database on Private Subnet:

MariaDB is a popular open-source database server that is widely used by developers, database enthusiasts, and also in production environments. It's a fork of MySQL and has been preferred to MySQL since the takeover of MySQL by Oracle.

- **Step1)** Install MariaDB.



- **Step2)** Securing your MySQL server

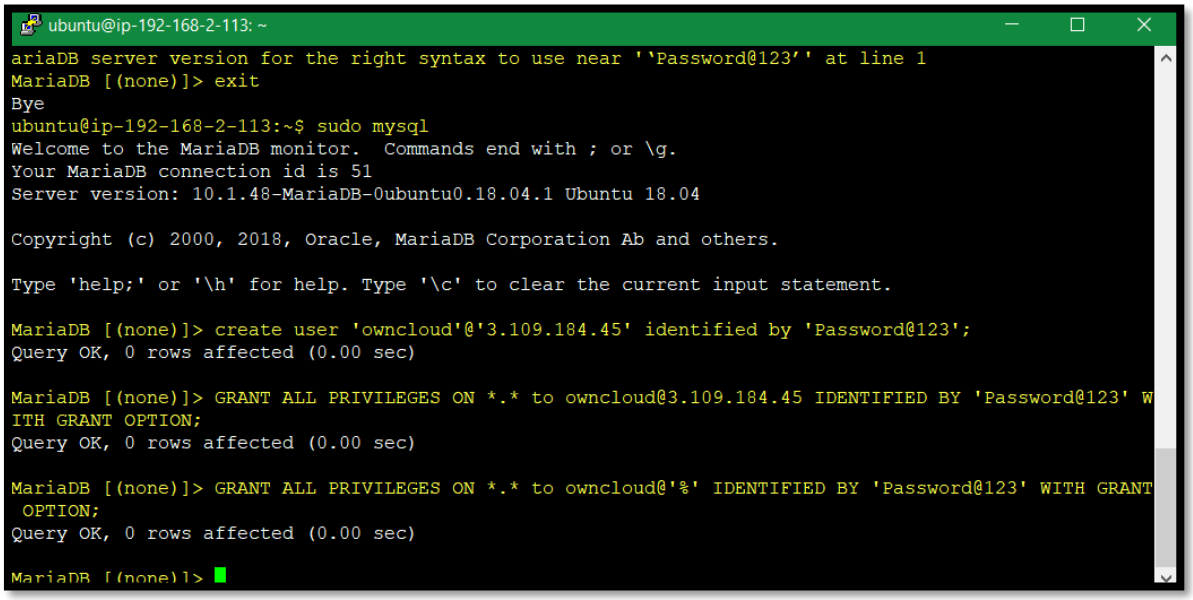
```
Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.

Set root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!


By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
 ... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
ubuntu@ip-192-168-2-113:~$ clear
```

❖ Creating ownCloud Database:

- **Step1)** We need to create a database for ownCloud to store files during and after installation. So, log in to MariaDB.

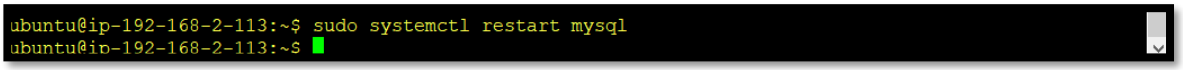
```
ubuntu@ip-192-168-2-113: ~
ubuntu@ip-192-168-2-113:~$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 49
Server version: 10.1.48-MariaDB-0ubuntu0.18.04.1 Ubuntu 18.04

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

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

MariaDB [(none)]>
```

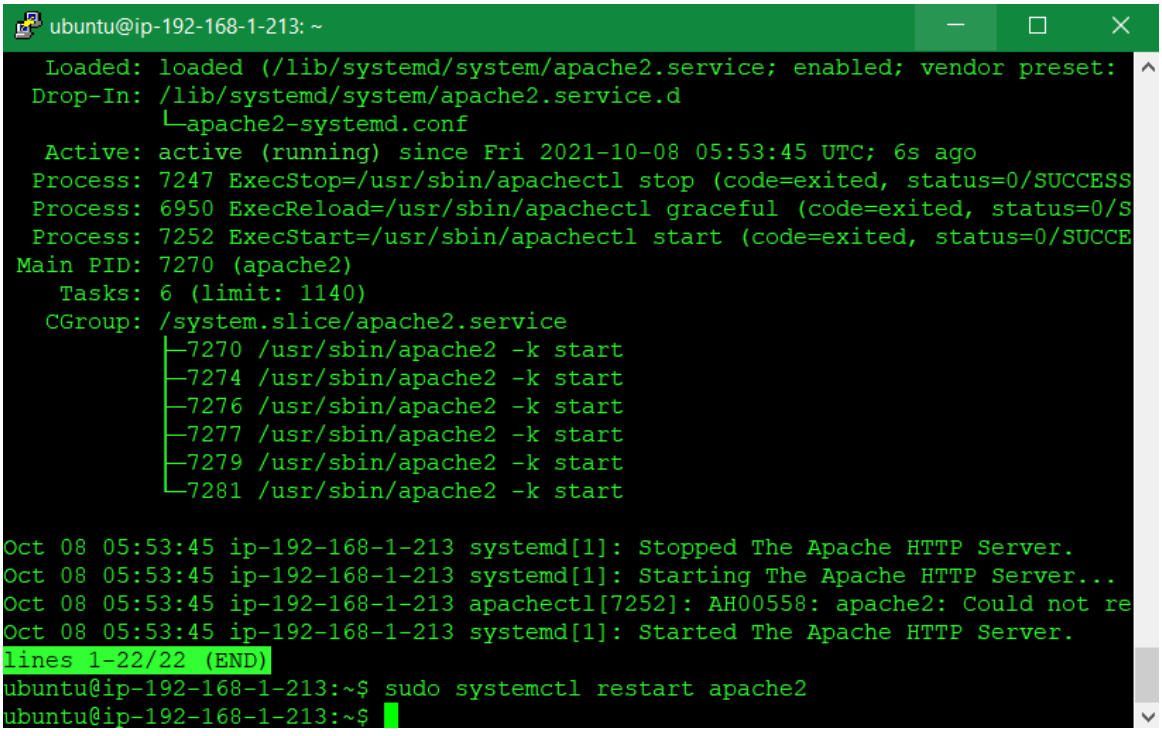
➢ **Step2)** Create user, database and grant the permissions.



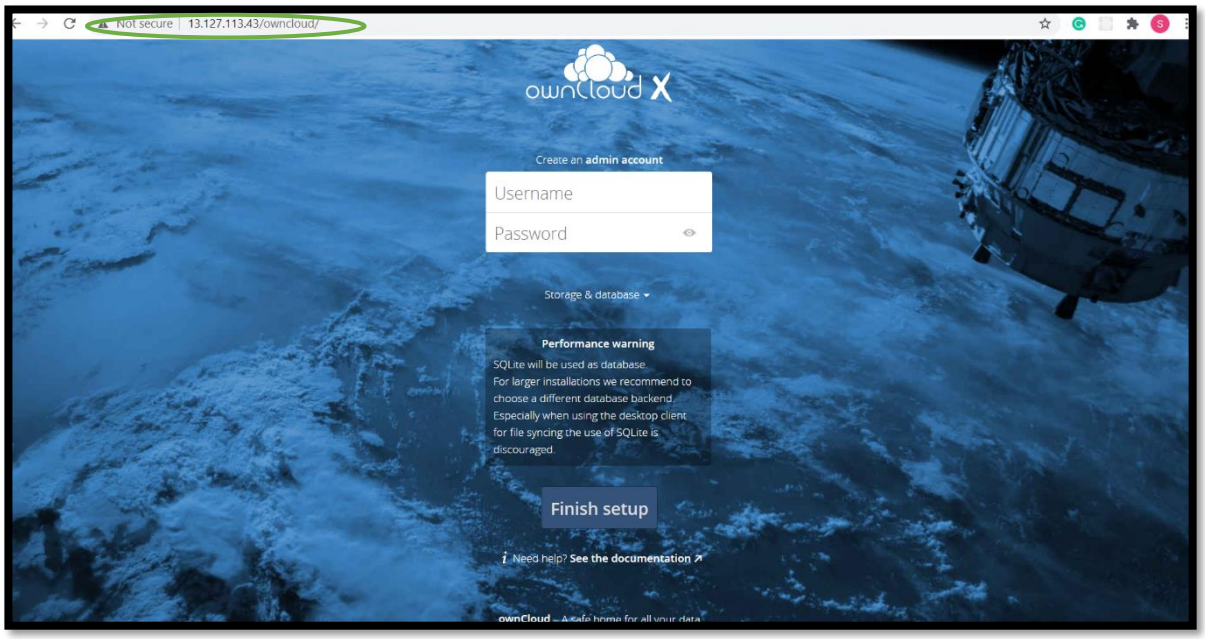
❖ Finalising ownCloud Installation:

Before opening our ownCloud in browser restart Apache in Public machine and MySql in private machine.

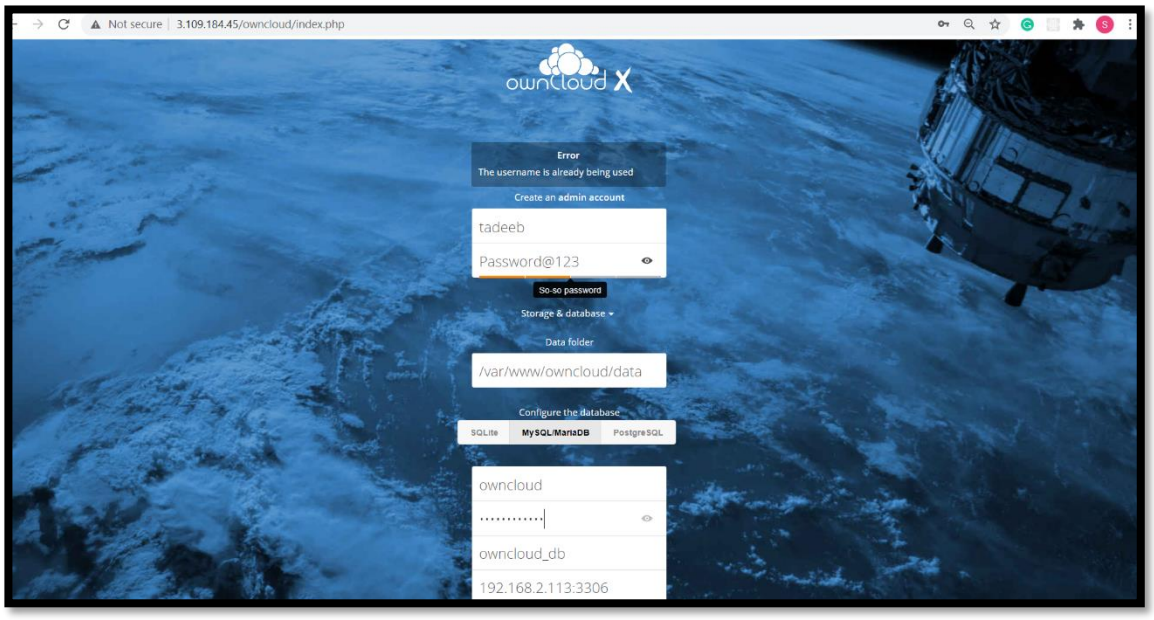




- **Step1)** With all the necessary configurations finalized, the only part remaining is to install OwnCloud on a browser. So, head out to your browser and type in your server's address followed by the /owncloud suffix.



- **Step2)** Just below, click on 'Storage and database'. Select 'MySQL / MariaDB' under the 'configure the database' section and fill in the database credentials that you defined whilst creating the database for OwnCloud i.e database user, password of the database user, & database name. Note enter Private address of private machine along with mysql port.

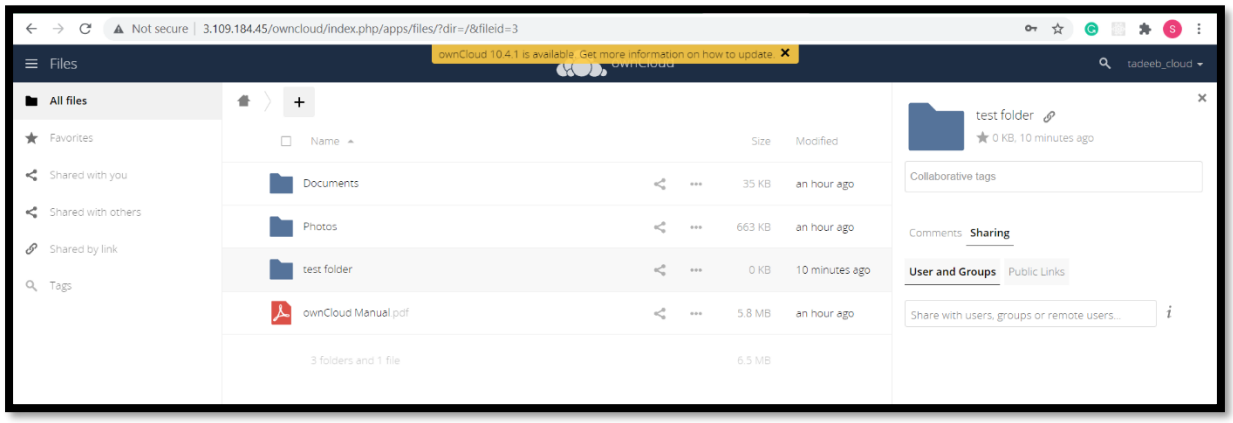


❖ Conclusion:

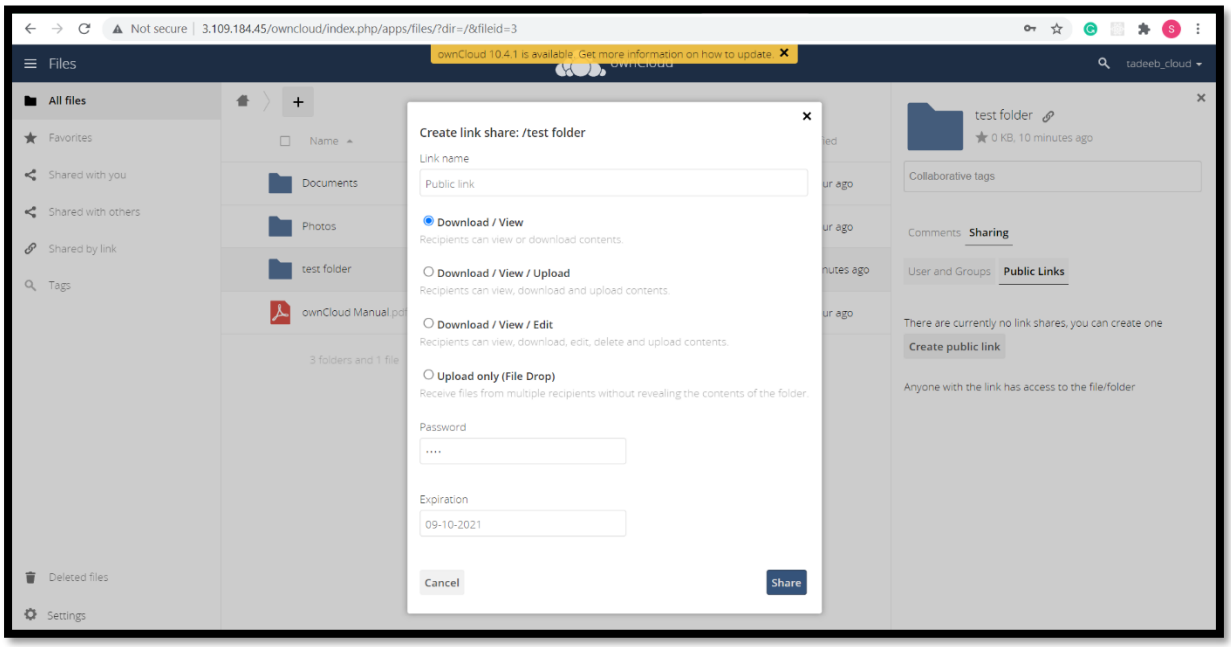
I was successfully able to log-in into ownCloud environment and share file or folders with other users of same enterprise i.e., belonging to same private subnet or right access.

Initially these are the default folders in my ownCloud:

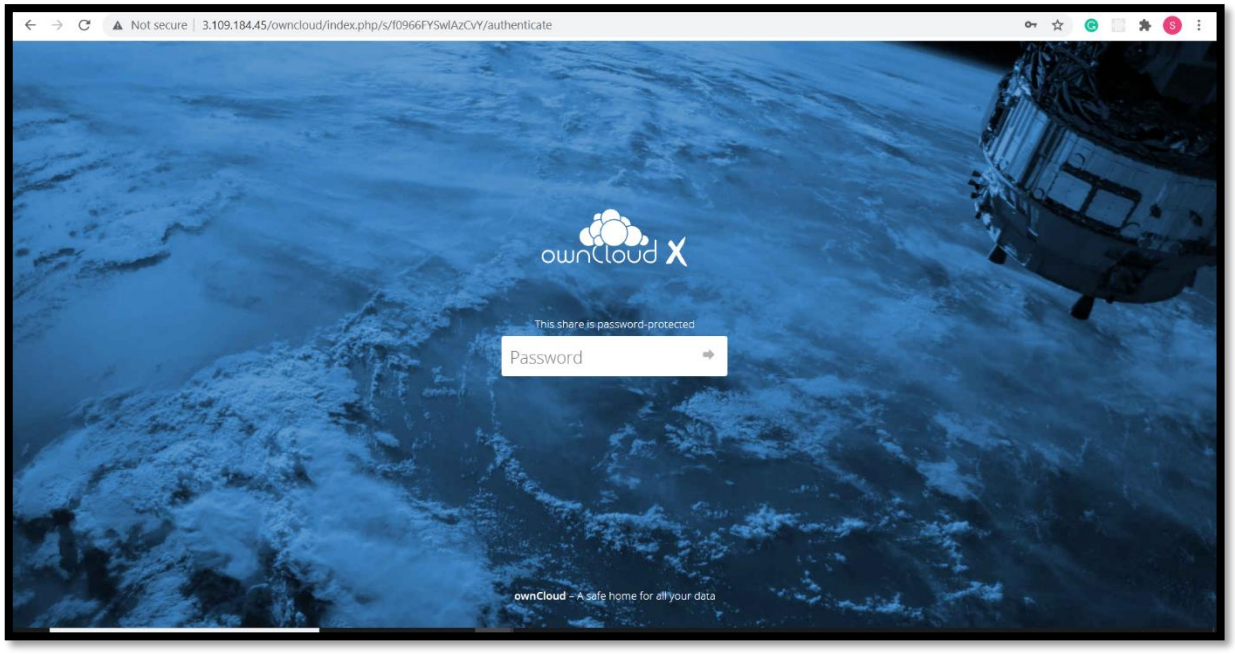
I created a "test folder" for sharing with other members of my own enterprise.



I then created password & expiry date along with various permissions on test folder.



Now I entered the link in the Browser and owncloud environment opened but it asked me password. If I enter correct password it will grant me access otherwise I wont be able to access the folder.



Hence we can conclude that our ownCloud is installed properly.