



# Discipline

ownCloud using AWS

CS5002

Course Project

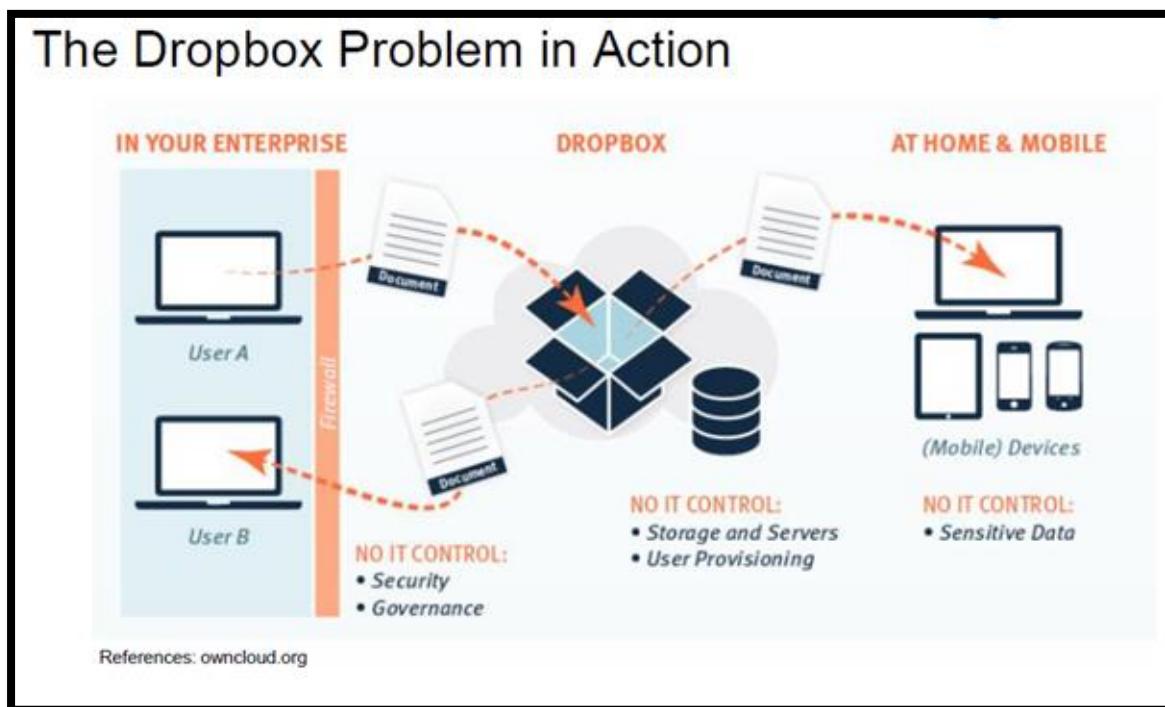
**Sheikh Muhammed Tadeeb (AU19B1014)**

---

## ❖ 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.

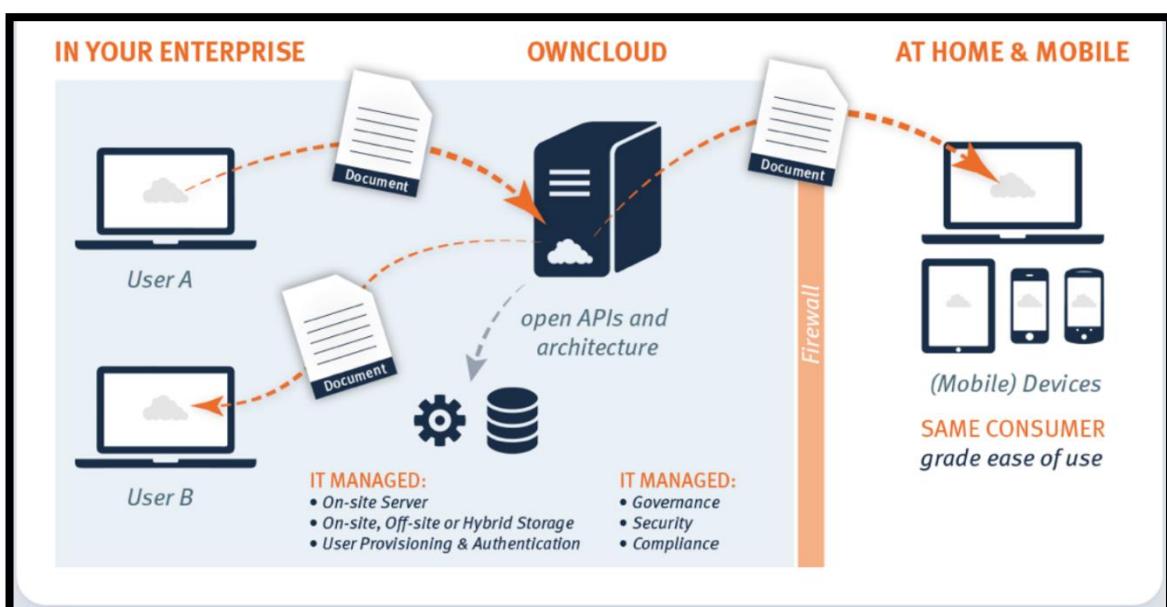
## ❖ Objective:

The following are the objectives of this project:

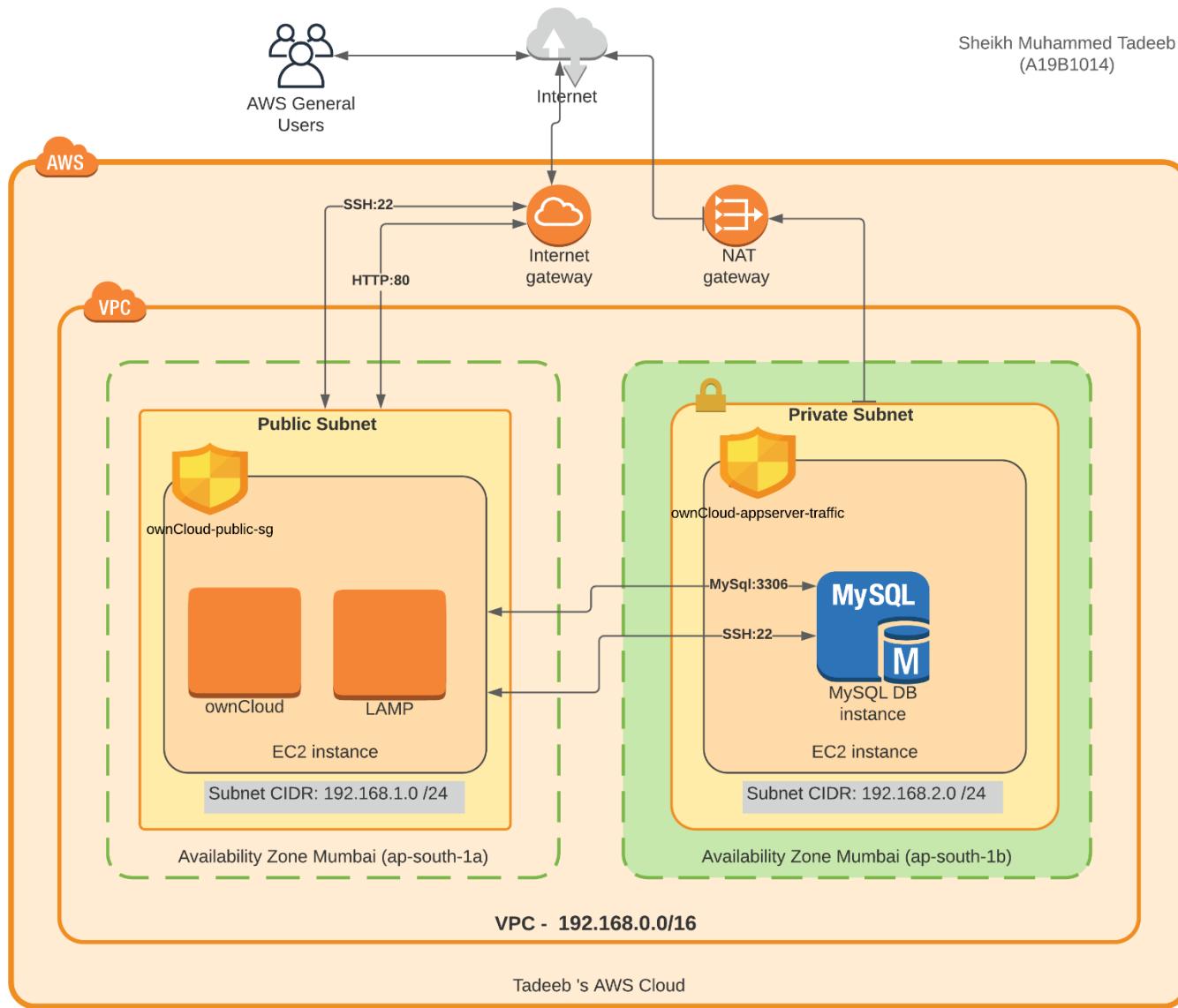
1. Implement the ownCloud solution for a small workgroup, which can cater up to 150 users by using various AWS services. The solution will be completely deployed on AWS. i.e., Implement 2 different subnets (one public and other private) in a custom VPC called owncloud-vpc.
2. The ownCloud app should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

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR
default	vpc-f99b4b92	Available	172.31.0.0/16	-

**VPC settings**

Name tag - optional  
Creates a tag with a key of 'Name' and a value that you specify.  
ownCloud-vpc-192.168.0.0/16

IPv4 CIDR block Info  
192.168.0.0/16

IPv6 CIDR block Info  
 No IPv6 CIDR block  
 Amazon-provided IPv6 CIDR block  
 IPv6 CIDR owned by me

Tenancy Info  
Default

**Tags**  
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
Q. Name	Q. ownCloud-vpc-192.168.0.0/16

Add new tag  
You can add 49 more tags.

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP options set
ownCloud-vpc-192.168.0.0/16	vpc-0571418781f270221	Available	192.168.0.0/16	-	dopt-80d16eeb
default	vpc-f99b4b92	Available	172.31.0.0/16	-	dopt-80d16eeb

**vpc-0571418781f270221 / ownCloud-vpc-192.168.0.0/16**

**Details**

VPC ID vpc-0571418781f270221	State Available	DNS hostnames Disabled	DNS resolution Enabled
Tenancy Default	DHCP options set dopt-80d16eeb	Main route table rtb-07dfaae34d2c9e47	Main network ACL acl-0683c71a34dd02645

## ❖ 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.

The screenshot shows the AWS VPC 'Create subnet' wizard. In the 'VPC' section, the VPC ID is selected as 'vpc-0571418781f270221 (ownCloud-vpc-192.168.0.0/16)'. Under 'Associated VPC CIDRs', the IPv4 CIDR is listed as '192.168.0.0/16'. The 'Subnet settings' section contains the following details:

- Subnet 1 of 1**:
  - Subnet name**: 'ownCloud-public-192.168.1.0/24' (input field)
  - A note states: 'The name can be up to 256 characters long.'
  - Availability Zone**: 'Asia Pacific (Mumbai) / ap-south-1a' (dropdown menu)
  - IPv4 CIDR block**: '192.168.1.0/24' (input field)
  - Tags - optional**: A single tag is defined with Key 'Name' and Value 'ownCloud-public-192.168.1.0/'.
  - Add new tag** and **Remove** buttons are available for managing tags.
  - A note says: 'You can add 49 more tags.'
  - Add new subnet** button is located at the bottom of the subnet configuration area.

At the bottom right of the wizard, there are 'Cancel' and 'Create subnet' buttons.

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

The screenshot shows the 'Modify auto-assign IP settings' page for a specific subnet. The 'Subnet ID' is listed as 'subnet-0e02f0ea1fc87feaa'. Under 'Auto-assign IPv4', the 'Enable auto-assign public IPv4 address' checkbox is checked. Under 'Auto-assign customer-owned IPv4 address', the checkbox is unchecked with a note: 'Option disabled because no customer owned pools found.' At the bottom right are 'Cancel' and 'Save' buttons.

The screenshot shows the 'Subnets (1/1)' list page. A green banner at the top indicates 'You have successfully modified auto-assign IP settings.' The 'Public IPv4 address' setting is shown as 'Yes'. The 'Auto-assign public IPv4 address' setting is also highlighted with a green oval. Below the table, the 'Details' tab is selected, showing the subnet's configuration: Subnet ID is 'subnet-0e02f0ea1fc87feaa', Subnet ARN is 'arnaws:ec2:ap-south-1:137496460515:subnet/subnet-0e02f0ea1fc87feaa', State is 'Available', and IPv4 CIDR is '192.168.1.0/24'. Other tabs include 'Flow logs', 'Route table', 'Network ACL', 'CIDR reservations', 'Sharing', and 'Tags'.

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

AWS 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**

Q 192.168.2.0/24 X

**Tags - optional**

Key	Value - optional
Q Name X	Q ownCloud-private-192.168.2.0 X Remove

Add new tag

You can add 49 more tags.

Remove

Add new subnet

Cancel Create subnet

New VPC Experience  
Tell us what you think

Services ▾

Search for services, features, marketplace products, and docs [Alt+S]

administrator @ 1374-9646-0515 Mumbai Support

You have successfully created 1 subnet: subnet-02e1f06b1a461c180

Subnets (1/2) Info

Filter subnets

VPC: vpc-0571418781f270221 X Clear filters

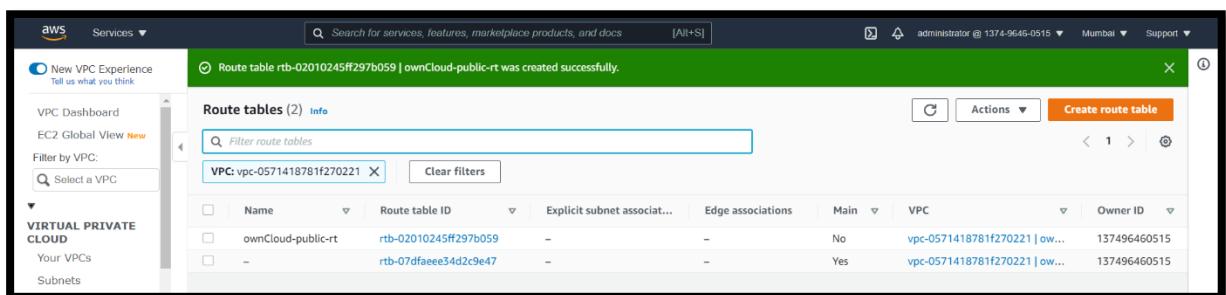
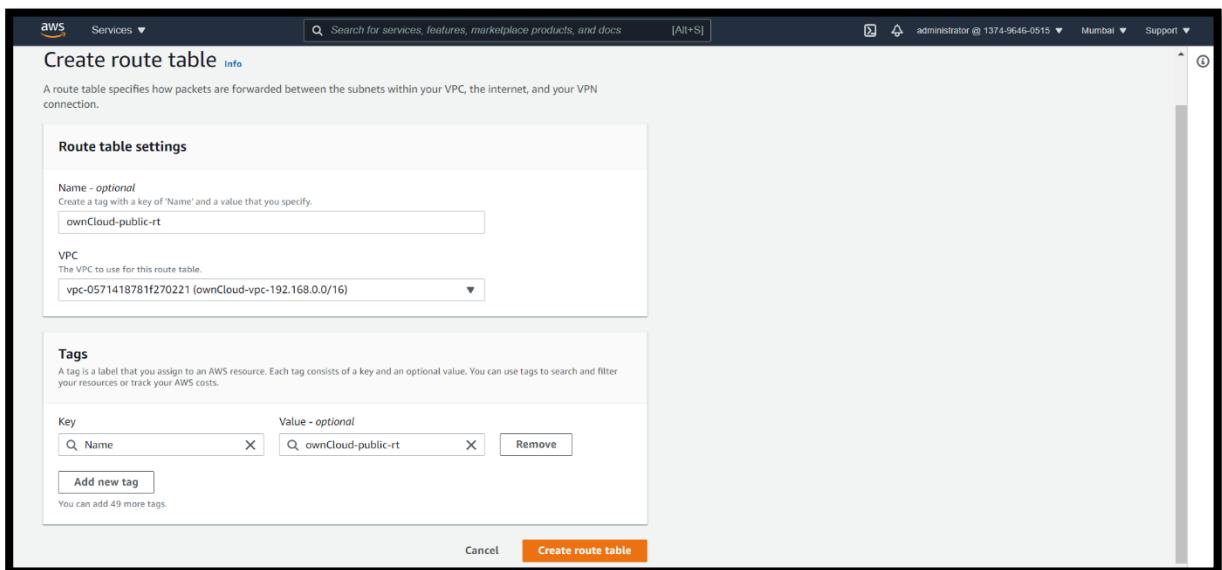
Name	Subnet ID	State	VPC	IPv4 CIDR
ownCloud-public-192.168.1.0/24	subnet-0e02f0ea1fc87feaa	Available	vpc-0571418781f270221   ow...	192.168.1.0/24
ownCloud-private-192.168.2.0/24	subnet-02e1f06b1a461c180	Available	vpc-0571418781f270221   ow...	192.168.2.0/24

subnet-02e1f06b1a461c180 / ownCloud-private-192.168.2.0/24

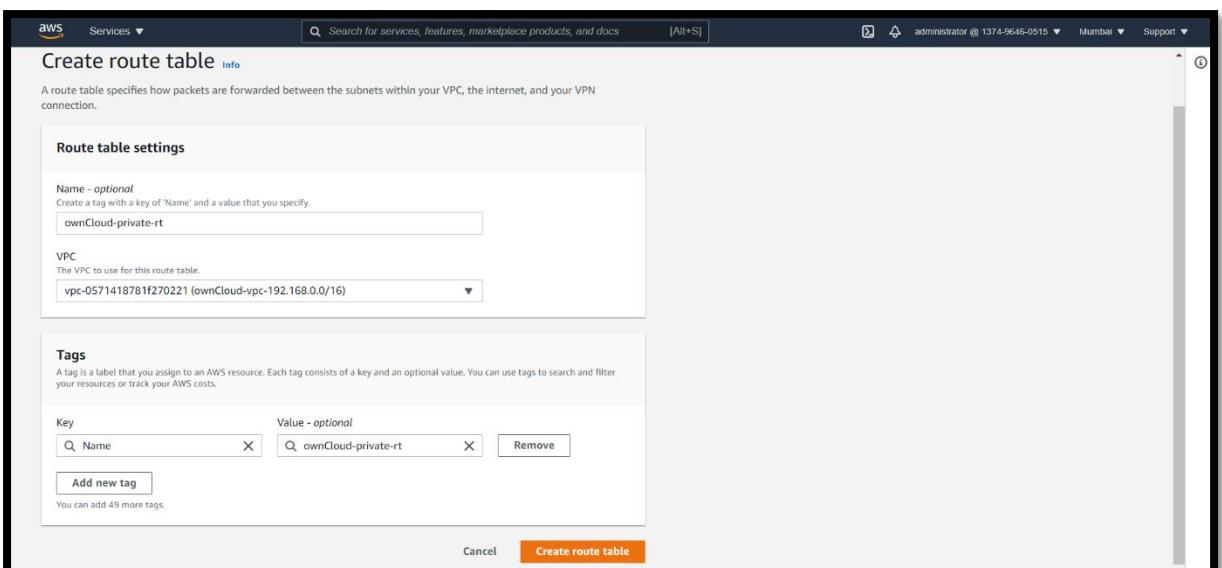
Details Flow logs Route table Network ACL CIDR reservations Sharing Tags

## ❖ 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.



The screenshot shows the AWS VPC Route Tables page. The left sidebar has 'Route Tables' selected under 'Virtual Private Cloud'. The main area displays a table titled 'Route tables (1/3)'. The table has columns for Name, Route table ID, Explicit subnet associat..., Edge associations, Main, VPC, and Owner ID. There are three entries:

Name	Route table ID	Explicit subnet associat...	Edge associations	Main	VPC	Owner ID
ownCloud-private-rt	rtb-08bb0236711a4b3b0	-	-	No	vpc-0571418781f270221   ow...	137496460515
ownCloud-public-rt	rtb-02010245ff297b059	-	-	No	vpc-0571418781f270221   ow...	137496460515
-	rtb-07dfae34d2c9e47	-	-	Yes	vpc-0571418781f270221   ow...	137496460515

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

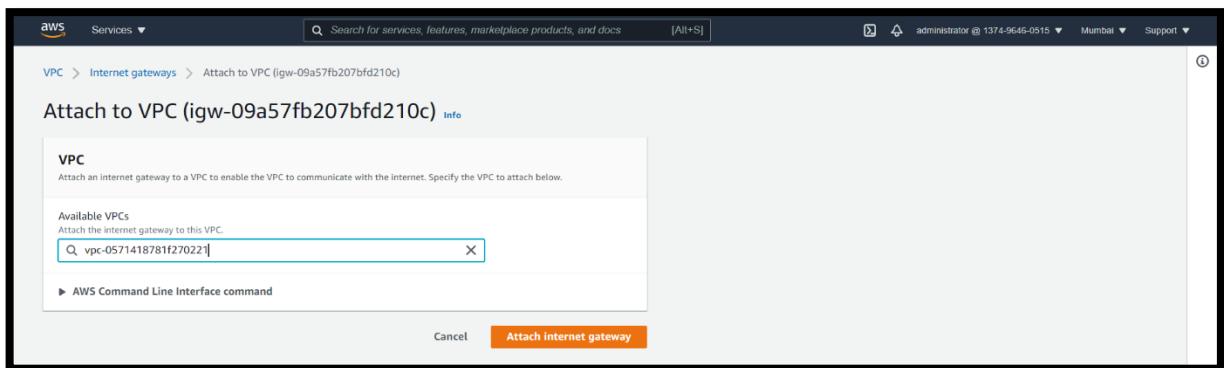
The screenshot shows the 'Create internet gateway' wizard. The 'Internet gateway settings' step is displayed. It includes a 'Name tag' field containing 'ownCloud-igw' and a 'Tags - optional' section with one tag 'Name: ownCloud-igw'. At the bottom are 'Cancel' and 'Create internet gateway' buttons.

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

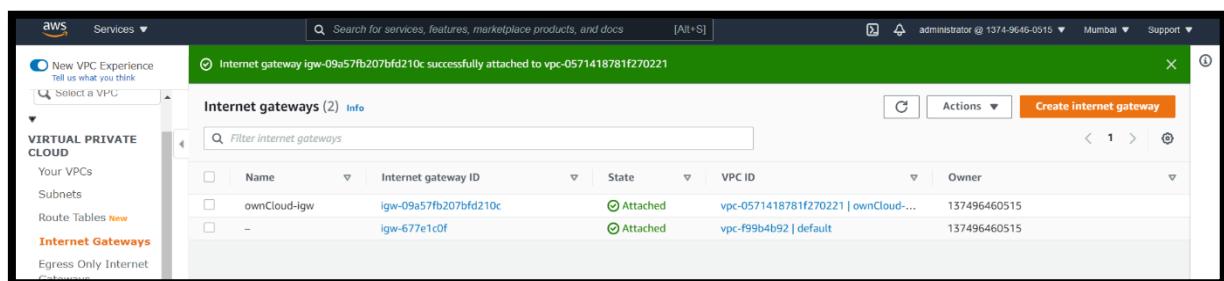
The screenshot shows the 'Internet gateways' page. The left sidebar has 'Internet Gateways' selected under 'Virtual Private Cloud'. The main area displays a table titled 'Internet gateways (2)'. The table has columns for Name, Internet gateway ID, State, VPC ID, and Owner. There are two entries:

Name	Internet gateway ID	State	VPC ID	Owner
ownCloud-igw	igw-09a57fb207bfd210c	Detached	-	137496460515
-	igw-677e1c0f	Attached	vpc-f99b4b92   default	137496460515

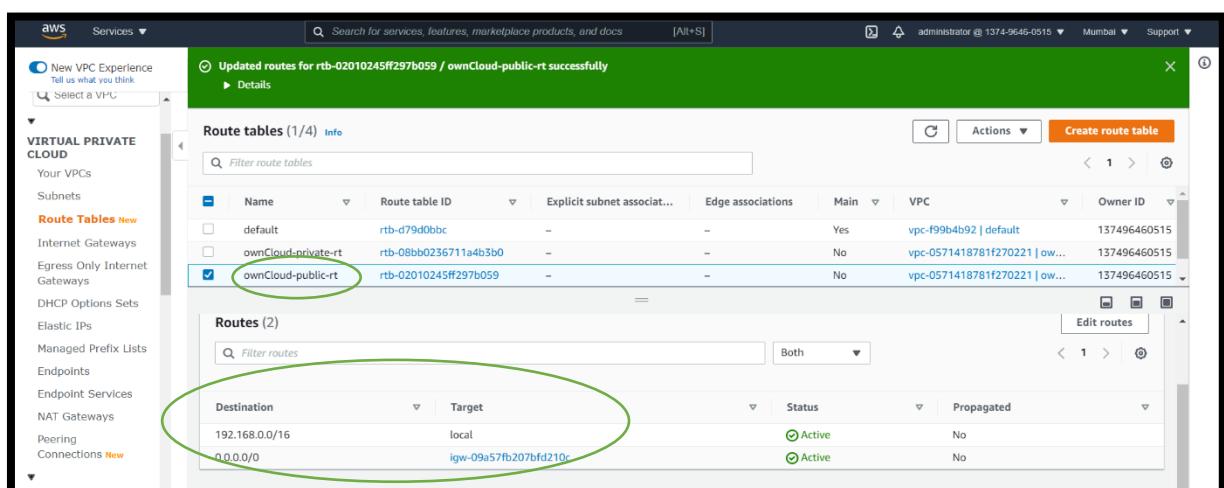
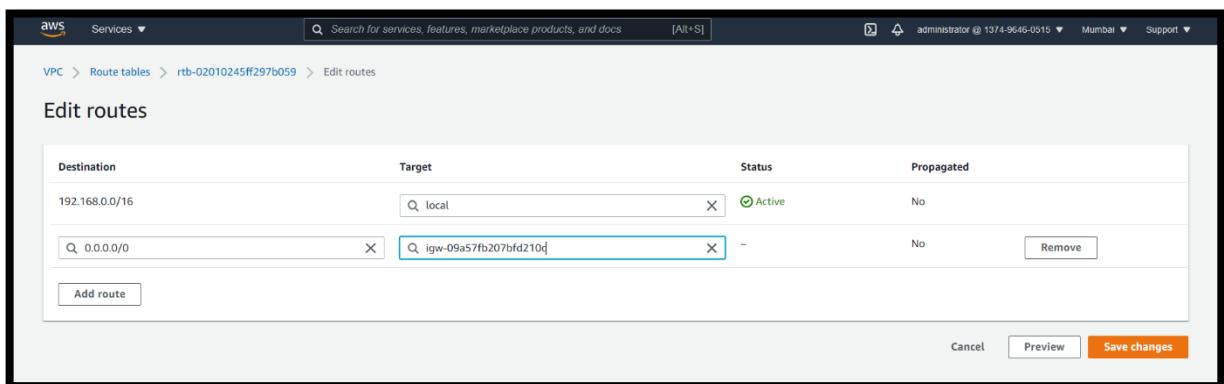
The screenshot shows the 'Internet gateways' page again. The 'Actions' dropdown for the 'ownCloud-igw' row shows an option to 'Attach to VPC'. This option is now grayed out, indicating the action is completed.



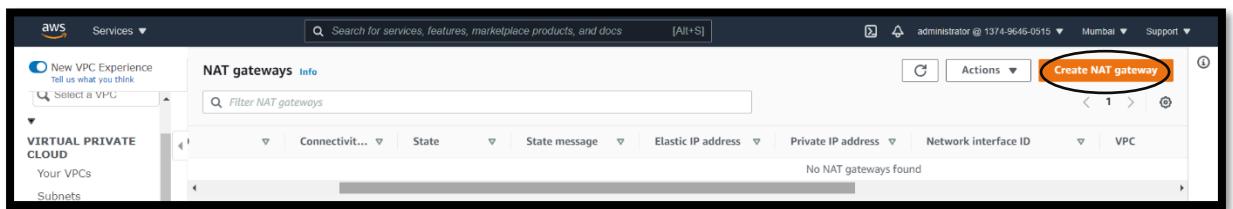
In the below image we can see that now our internet gateway is attached to our vpc.



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



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



The screenshot shows the AWS VPC NAT gateways page. At the top right, there is an orange button labeled "Create NAT gateway" which is circled in red.

**Create NAT gateway**

A highly available, managed Network Address Translation (NAT) service that instances in private subnets can use to connect to services in other VPCs, on-premises networks, or the internet.

### NAT gateway settings

**Name - optional**  
Create a tag with a key of 'Name' and a value that you specify.

The name can be up to 256 characters long.

**Subnet**  
Select a subnet in which to create the NAT gateway.

**Connectivity type**  
Select a connectivity type for the NAT gateway.  
 Public  
 Private

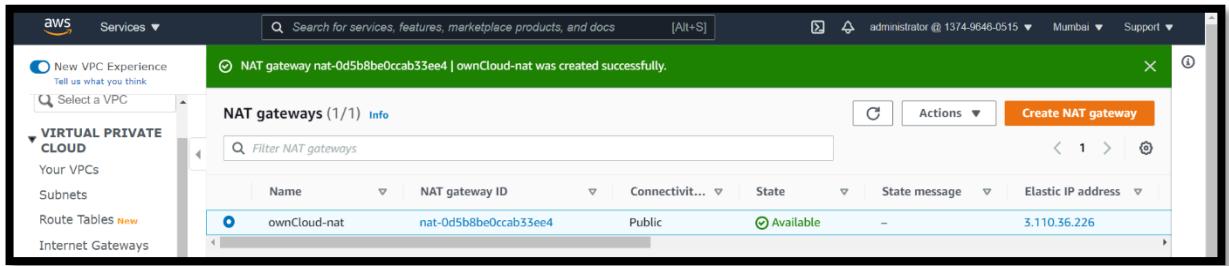
**Elastic IP allocation ID** [Info](#)  
Assign an Elastic IP address to the NAT gateway.  
 [Allocate Elastic IP](#)

### Tags

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	Remove
<input type="text" value="Name"/>	<input type="text" value="ownCloud-nat"/>	<a href="#">Remove</a>
<a href="#">Add new tag</a>		
You can add 49 more tags.		

[Cancel](#) [Create NAT gateway](#)



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

The screenshot shows the AWS VPC Route tables console. The 'Routes' tab is selected for the 'ownCloud-private-rt' route table. It contains one route entry:

Destination	Target	Status	Propagated
192.168.0.0/16	local	Active	No

The screenshot shows the 'Edit routes' dialog for the 'ownCloud-private-rt' route table. It displays two route entries:

Destination	Target	Status	Propagated
192.168.0.0/16	local	Active	No
0.0.0.0/0	nat-0d5b8be0ccb33ee4	-	No

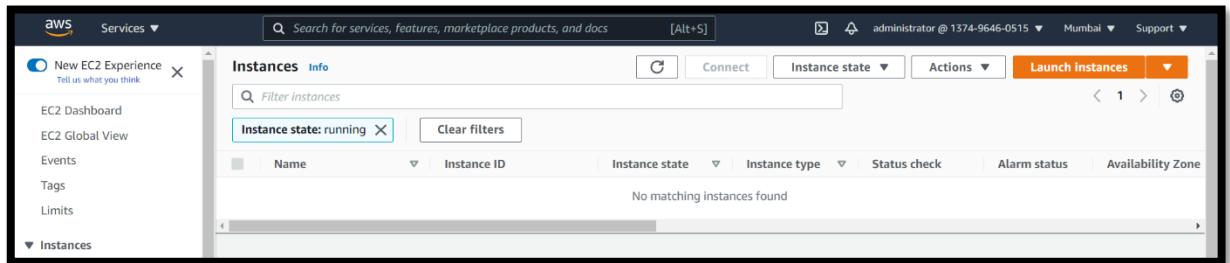
Buttons at the bottom include 'Add route', 'Cancel', 'Preview', and 'Save changes'.

The screenshot shows the AWS VPC Route tables console. A green banner at the top indicates that 'Updated routes for rtb-08bb0236711a4b3b0 / ownCloud-private-rt successfully'. The 'Routes' tab is selected for the 'ownCloud-private-rt' route table. It now shows two active routes:

Destination	Target	Status	Propagated
192.168.0.0/16	local	Active	No
0.0.0.0/0	nat-0d5b8be0ccb33ee4	Active	No

## ❖ Ec2 Instance Creation:

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



Selecting the Ubuntu 18.04 instance.

Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Search bar: Ubuntu

Quick Start (3)
My AMIs (0)
AWS Marketplace (865)
Community AMIs (14509)
<input checked="" type="checkbox"/> Free tier only
<b>Ubuntu Server 20.04 LTS (HVM), SSD Volume Type</b> - ami-0c1a7f89451184c8b (64-bit x86) / ami-0d18acc6e813fd2e0 (64-bit Arm)
Ubuntu Server 20.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical ( <a href="http://www.ubuntu.com/cloud/services">http://www.ubuntu.com/cloud/services</a> ). Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
<input checked="" type="checkbox"/> <b>Ubuntu Server 18.04 LTS (HVM), SSD Volume Type</b> - ami-04bde106886a53080 (64-bit x86) / ami-04f6f742e1d9012e3 (64-bit Arm)
Ubuntu Server 18.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical ( <a href="http://www.ubuntu.com/cloud/services">http://www.ubuntu.com/cloud/services</a> ). Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
<input checked="" type="checkbox"/> <b>Ubuntu Server 16.04 LTS (HVM), SSD Volume Type</b> - ami-0860c9429babaa6ad2 (64-bit x86) / ami-03ab5f3b31d5ee063 (64-bit Arm)
Ubuntu Server 16.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical ( <a href="http://www.ubuntu.com/cloud/services">http://www.ubuntu.com/cloud/services</a> ). Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance families ▾ Current generation ▾ Show/Hide Columns

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	<b>t2.micro</b> Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes
t2	t2.2xlarge	8	32	EBS only	-	Moderate	Yes

Cancel Previous Review and Launch Next: Configure Instance Details

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

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of Instances: 1

Purchasing option: Request Spot instances

Network: vpc-0571418781270221 | ownCloud-vpc-192.168.0.0/16

Subnet: subnet-0e02f0ea1fc87feaa | ownCloud-public-192.168.0.0/16

Auto-assign Public IP: Use subnet setting (Enable)

Placement group: Add instance to placement group

Capacity Reservation: Open

Domain join directory: No directory

IAM role: None

Shutdown behavior: Stop

Cancel Previous Review and Launch Next: Add Storage

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/sda1	snap-0c755833850745549	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

Add New Volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

Cancel Previous Review and Launch Next: Add Tags

Step 5: Add Tags

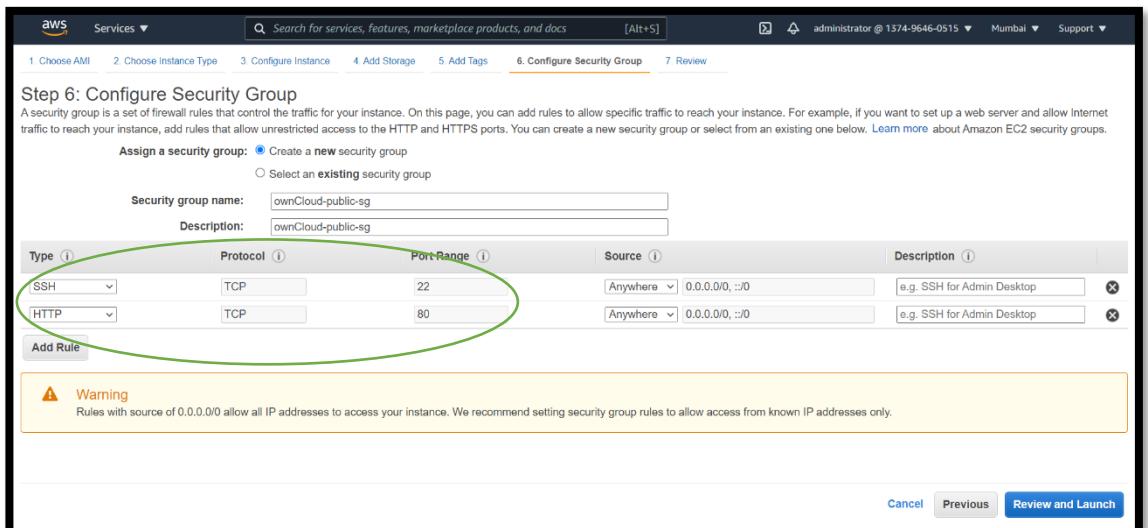
A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. A copy of a tag can be applied to volumes, instances or both. Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key (128 characters maximum)	Value (256 characters maximum)	Instances	Volumes	Network Interfaces
Name	ownCloud-public-machine	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Add another tag (Up to 50 tags maximum)

Cancel Previous Review and Launch Next: Configure Security Group

- **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.

Security Groups (2) <small>Info</small>							
<input type="button" value="Actions"/> <input type="button" value="Create security group"/>							
<input type="text" value="VPC ID: vpc-0571418781f270221"/> <input type="button" value="Clear filters"/>							
Name	Security group ID	Security group name	VPC ID	Description	Owner		
—	sg-031c467c5cbdd750	ownCloud-public-sg	vpc-0571418781f270221	ownCloud-public-sg	1374964		
—	sg-0769bc4c1073630a9	default	vpc-0571418781f270221	default VPC security gr...	1374964		

**Create security group Info**

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

**Basic details**

Security group name Info  
ownCloud-appserver-traffic  
Name cannot be edited after creation.

Description Info  
ownCloud-appserver-traffic

VPC Info  
vpc-0571418781f270221

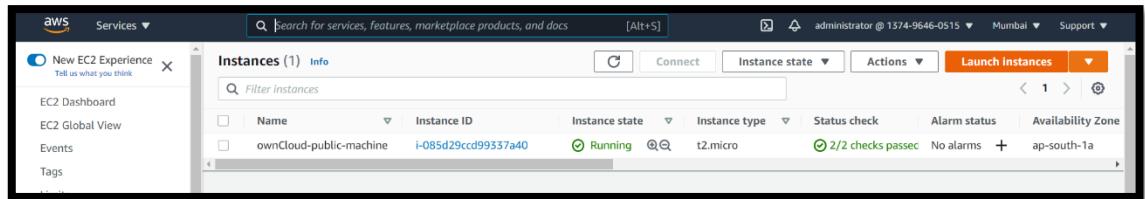
The screenshot shows the AWS Management Console interface for creating a new security group. The security group is named 'ownCloud-appserver-traffic'. In the 'Inbound rules' section, there are two entries: one for SSH (port 22) from '192.168.1.0/24' and another for MySQL/Aurora (port 3306) from '192.168.1.0/24'. Both rules have 'Delete' buttons next to them. In the 'Outbound rules' section, there is a single entry for 'All traffic' to '0.0.0.0/0'. The 'Tags - optional' section indicates no tags are associated with the resource. At the bottom right, there are 'Cancel' and 'Create security group' buttons.

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

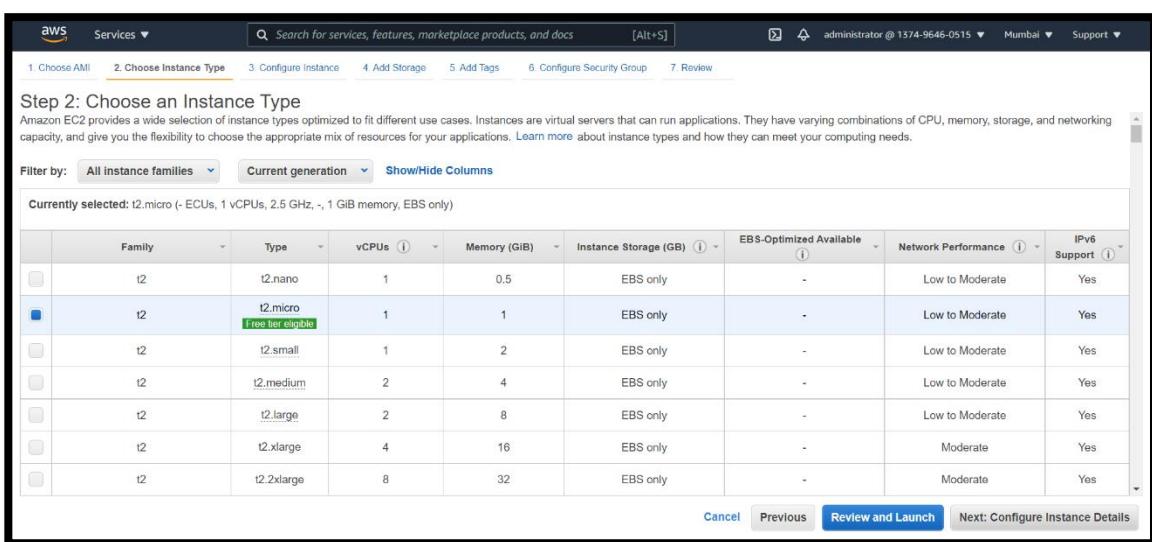
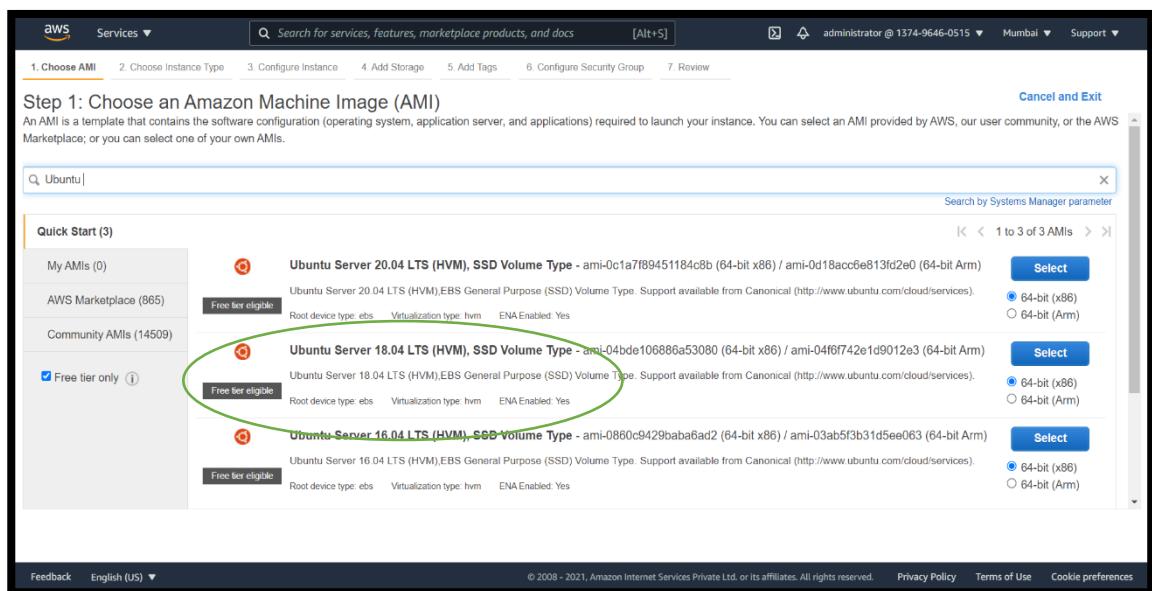
The screenshot shows the AWS Management Console interface for viewing security groups. The 'ownCloud-appserver-traffic' security group is selected, highlighted with a green oval. The 'Inbound rules' table shows the two rules defined earlier: SSH (port 22) and MySQL/Aurora (port 3306) both originating from '192.168.1.0/24'. The 'Actions' button is visible at the top right of the main content area.

Name	Security group ID	Security group name	VPC ID	Description
-	sg-031c467c3cbcd750	ownCloud-public-sg	vpc-0571418781f270221	ownCloud-public-sg
-	sg-0769bc4c1073630a9	default	vpc-0571418781f270221	default VPC security gr...
<input checked="" type="checkbox"/>	sg-0aaf2608282f0136f	ownCloud-appserver-traffic	vpc-0571418781f270221	ownCloud-appserver-t...

- **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.



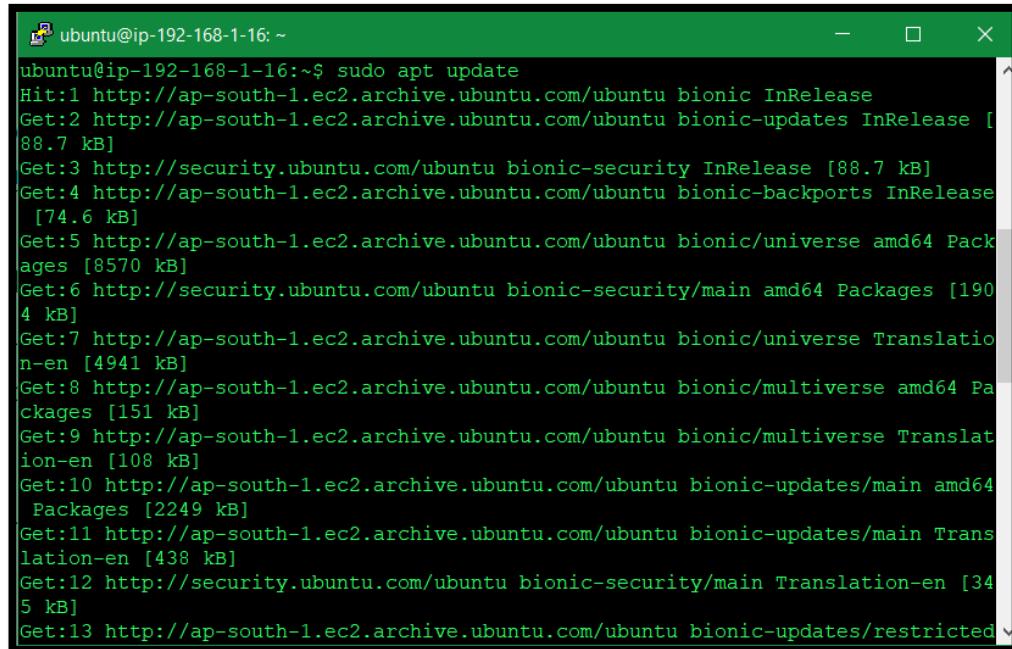
Selecting our ownCloud-vpc-192.168.0.0/16 in Network and Subnet as public subnet.

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.

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	192.168.1.0/24	
MySQL/Aurora	TCP	3306	192.168.1.0/24	

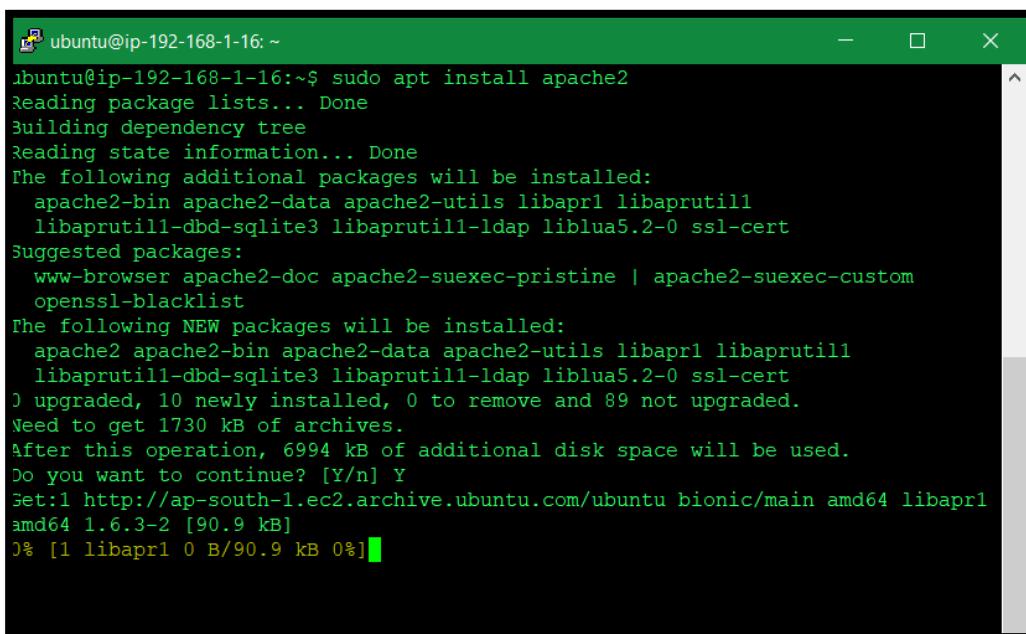
## ❖ Installing Apache and Updating Firewall (if required):

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



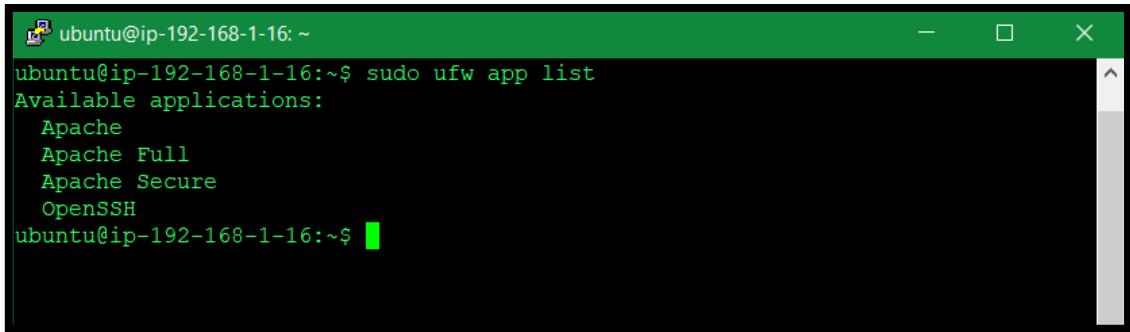
```
ubuntu@ip-192-168-1-16:~$ sudo apt update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [8570 kB]
Get:6 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [1904 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu bionic/universe Translation-en [4941 kB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [151 kB]
Get:9 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu bionic/multiverse Translation-en [108 kB]
Get:10 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [2249 kB]
Get:11 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu bionic-updates/main Translation-en [438 kB]
Get:12 http://security.ubuntu.com/ubuntu bionic-security/main Translation-en [345 kB]
Get:13 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu bionic-updates/restricted
```

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



```
ubuntu@ip-192-168-1-16:~$ sudo apt install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.2-0 ssl-cert
Suggested packages:
  www-browser apache2-doc apache2-suexec-pristine | apache2-suexec-custom
  openssl-blacklist
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.2-0 ssl-cert
0 upgraded, 10 newly installed, 0 to remove and 89 not upgraded.
Need to get 1730 kB of archives.
After this operation, 6994 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu bionic/main amd64 libapr1
  amd64 1.6.3-2 [90.9 kB]
0% [1 libapr1 0 B/90.9 kB 0%]
```

- **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`



```
ubuntu@ip-192-168-1-16: ~
ubuntu@ip-192-168-1-16:~$ sudo ufw app list
Available applications:
 Apache
 Apache Full
 Apache Secure
 OpenSSH
ubuntu@ip-192-168-1-16:~$
```

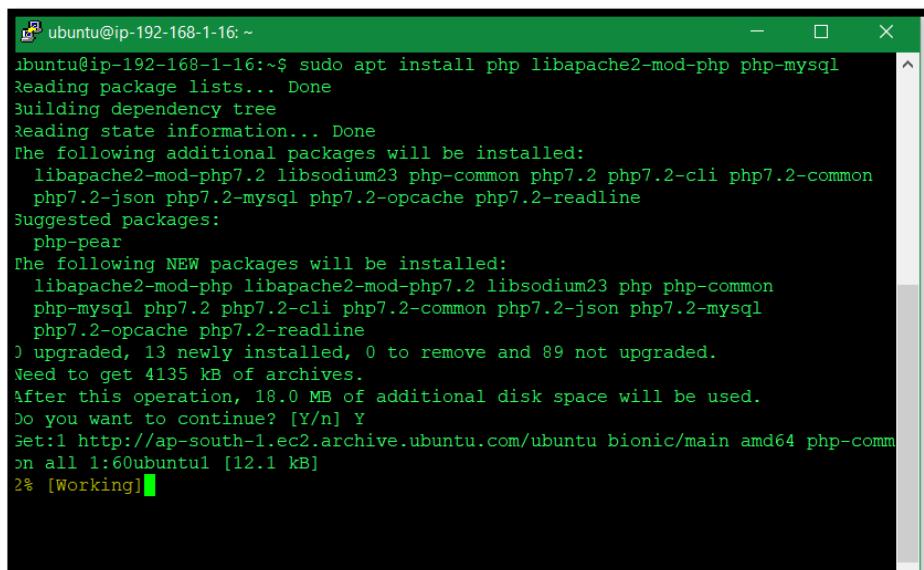
In above output Apache Full is there that means ports 80 and 443 are enabled.

## ❖ 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`

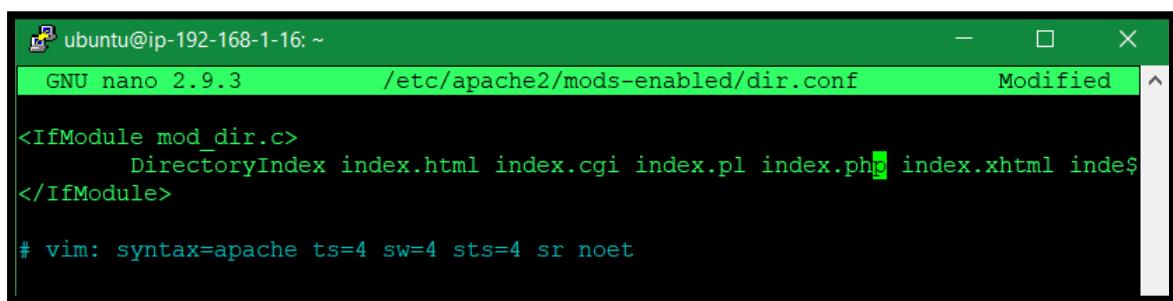


```
ubuntu@ip-192-168-1-16: ~
ubuntu@ip-192-168-1-16:~$ sudo apt install php libapache2-mod-php php-mysql
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libapache2-mod-php7.2 libsodium23 php-common php7.2 php7.2-cli php7.2-common
    php7.2-json php7.2-mysql php7.2-opcache php7.2-readline
Suggested packages:
  php-pear
The following NEW packages will be installed:
  libapache2-mod-php libapache2-mod-php7.2 libsodium23 php php-common
    php-mysql php7.2 php7.2-cli php7.2-common php7.2-json php7.2-mysql
    php7.2-opcache php7.2-readline
0 upgraded, 13 newly installed, 0 to remove and 89 not upgraded.
Need to get 4135 kB of archives.
After this operation, 18.0 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu bionic/main amd64 php-common all 1:60ubuntu1 [12.1 kB]
2% [Working]
```

## ❖ 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.

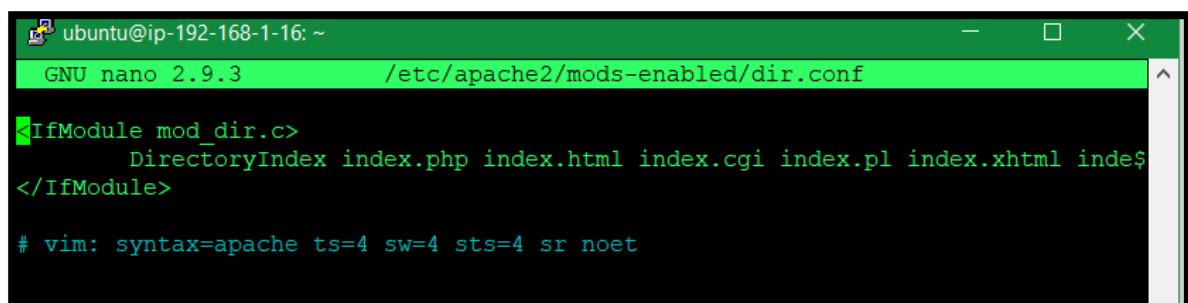
### Before:



```
ubuntu@ip-192-168-1-16: ~
GNU nano 2.9.3          /etc/apache2/mods-enabled/dir.conf      Modified
<IfModule mod_dir.c>
    DirectoryIndex index.html index.cgi index.pl index.php index.xhtml index.shtml
</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

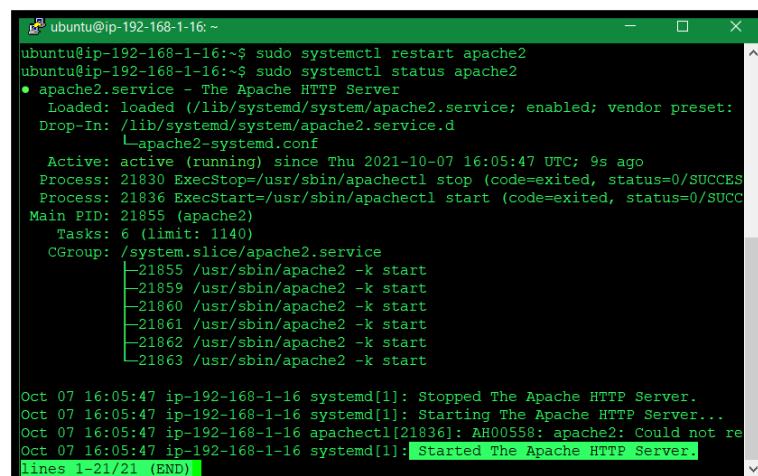
### After:



```
ubuntu@ip-192-168-1-16: ~
GNU nano 2.9.3          /etc/apache2/mods-enabled/dir.conf      Modified
<IfModule mod_dir.c>
    DirectoryIndex index.php index.html index.cgi index.pl index.xhtml index.shtml
</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

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



```
ubuntu@ip-192-168-1-16:~$ sudo systemctl restart apache2
ubuntu@ip-192-168-1-16:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: Drop-In: /lib/systemd/system/apache2.service.d
             └─apache2-systemd.conf
   Active: active (running) since Thu 2021-10-07 16:05:47 UTC; 9s ago
     Process: 21830 ExecStop=/usr/sbin/apachectl stop (code=exited, status=0/SUCCESS)
     Process: 21836 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 21855 (apache2)
      Tasks: 6 (limit: 1140)
        CGroub: /system.slice/apache2.service
                  ├─21855 /usr/sbin/apache2 -k start
                  ├─21859 /usr/sbin/apache2 -k start
                  ├─21860 /usr/sbin/apache2 -k start
                  ├─21861 /usr/sbin/apache2 -k start
                  ├─21862 /usr/sbin/apache2 -k start
                  └─21863 /usr/sbin/apache2 -k start

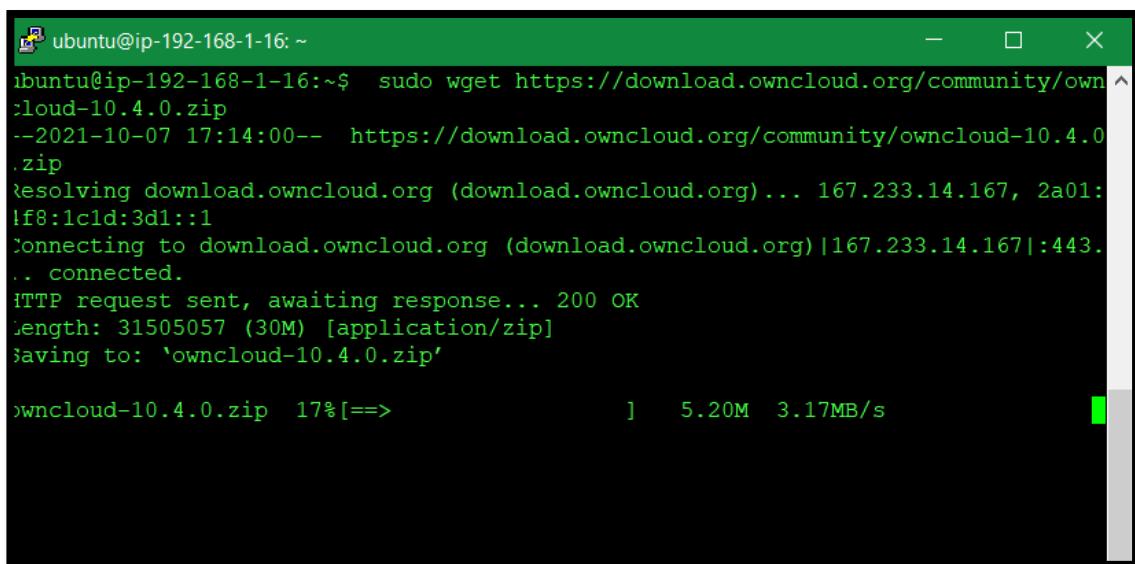
Oct 07 16:05:47 ip-192-168-1-16 systemd[1]: Stopped The Apache HTTP Server.
Oct 07 16:05:47 ip-192-168-1-16 systemd[1]: Starting The Apache HTTP Server...
Oct 07 16:05:47 ip-192-168-1-16 apachectl[21836]: AH00558: apache2: Could not re
Oct 07 16:05:47 ip-192-168-1-16 systemd[1]: Started The Apache HTTP Server.
lines 1-21/21 (END)
```

## ❖ 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:

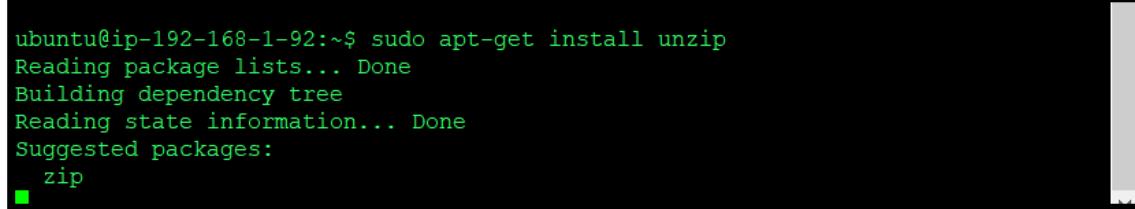
- Step1) Downloading zip file of ownCloud on Ubuntu machine.



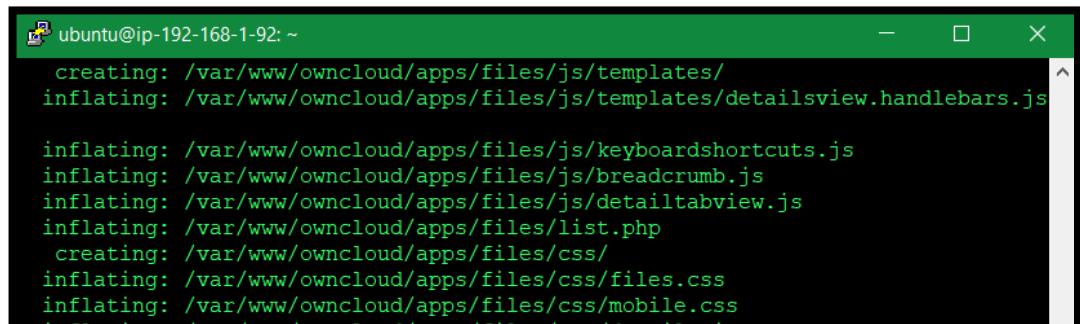
```
ubuntu@ip-192-168-1-16: ~
ubuntu@ip-192-168-1-16:~$ sudo wget https://download.owncloud.org/community/owncloud-10.4.0.zip
--2021-10-07 17:14:00-- https://download.owncloud.org/community/owncloud-10.4.0.zip
Resolving download.owncloud.org (download.owncloud.org)... 167.233.14.167, 2a01:lf8:1c1d:3d1::1
Connecting to download.owncloud.org (download.owncloud.org)|167.233.14.167|:443...
... connected.
HTTP request sent, awaiting response... 200 OK
Length: 31505057 (30M) [application/zip]
Saving to: 'owncloud-10.4.0.zip'

owncloud-10.4.0.zip 17%[==>]      5.20M  3.17MB/s
```

- Step2) Once downloaded, unzip the zipped package to the /var/www/ directory, but 1<sup>st</sup> we need to install unzip package.



```
ubuntu@ip-192-168-1-92: ~
ubuntu@ip-192-168-1-92:~$ sudo apt-get install unzip
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  zip
■
```



```
ubuntu@ip-192-168-1-92: ~
  creating: /var/www/owncloud/apps/files/js/templates/
  inflating: /var/www/owncloud/apps/files/js/templates/detailsview.handlebars.js

  inflating: /var/www/owncloud/apps/files/js/keyboardshortcuts.js
  inflating: /var/www/owncloud/apps/files/js/breadcrumb.js
  inflating: /var/www/owncloud/apps/files/js/detailtabview.js
  inflating: /var/www/owncloud/apps/files/list.php
  creating: /var/www/owncloud/apps/files/css/
  inflating: /var/www/owncloud/apps/files/css/files.css
  inflating: /var/www/owncloud/apps/files/css/mobile.css
```

## ❖ Configuring Apache for ownCloud:

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

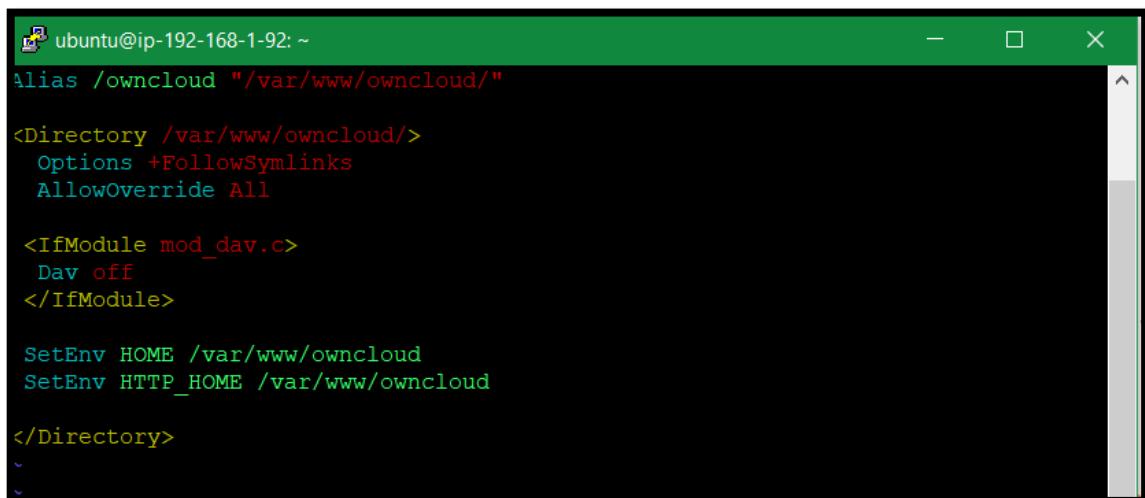
```
Alias /owncloud "/var/www/owncloud/"

<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>
```



A screenshot of a terminal window titled "ubuntu@ip-192-168-1-92: ~". The window contains the Apache configuration code shown in the previous block. The code is color-coded: Aliases and Directories are in red, Options and AllowOverride are in blue, and IfModule blocks are in green. The terminal has a dark background with light-colored text.

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

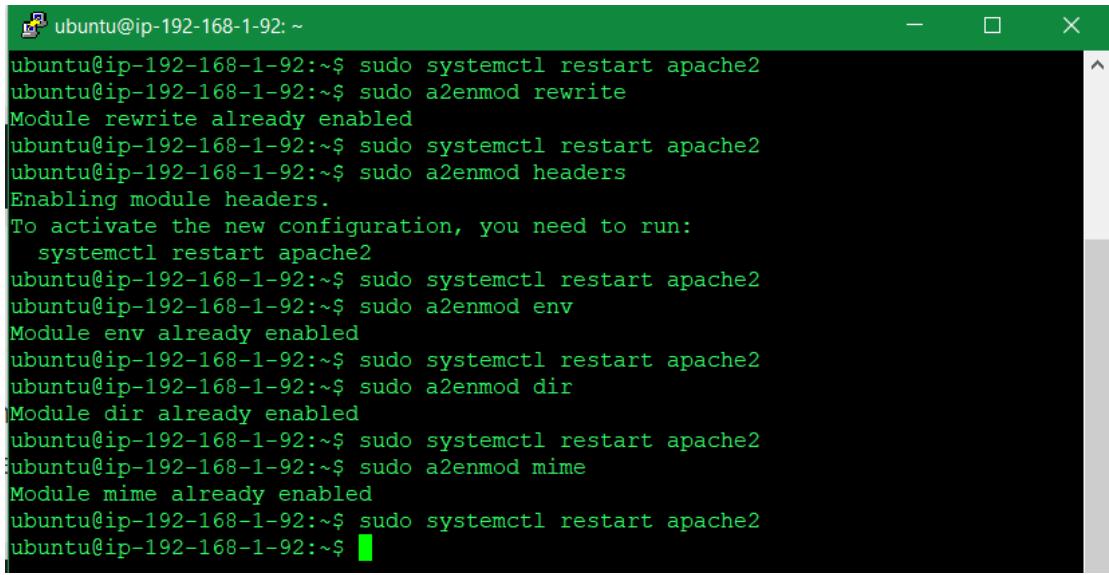
<Directory /var/www/owncloud/>
    Options +FollowSymlinks
    AllowOverride All

    <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.

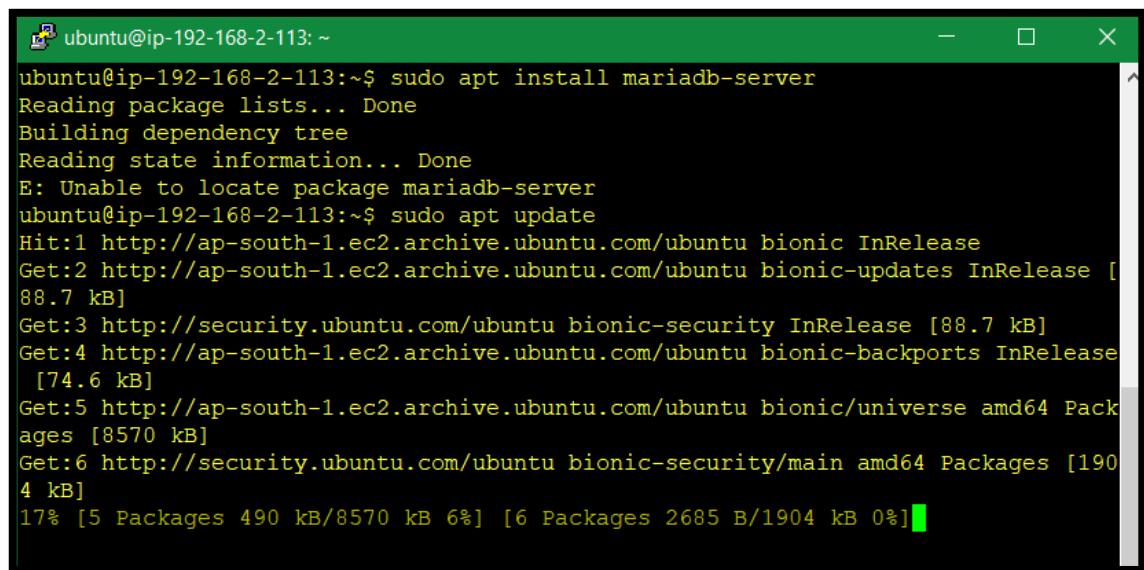


```
ubuntu@ip-192-168-1-92:~$ sudo systemctl restart apache2
ubuntu@ip-192-168-1-92:~$ sudo a2enmod rewrite
Module rewrite already enabled
ubuntu@ip-192-168-1-92:~$ sudo systemctl restart apache2
ubuntu@ip-192-168-1-92:~$ sudo a2enmod headers
Enabling module headers.
To activate the new configuration, you need to run:
    systemctl restart apache2
ubuntu@ip-192-168-1-92:~$ sudo systemctl restart apache2
ubuntu@ip-192-168-1-92:~$ sudo a2enmod env
Module env already enabled
ubuntu@ip-192-168-1-92:~$ sudo systemctl restart apache2
ubuntu@ip-192-168-1-92:~$ sudo a2enmod dir
Module dir already enabled
ubuntu@ip-192-168-1-92:~$ sudo systemctl restart apache2
ubuntu@ip-192-168-1-92:~$ sudo a2enmod mime
Module mime already enabled
ubuntu@ip-192-168-1-92:~$ sudo systemctl restart apache2
ubuntu@ip-192-168-1-92:~$
```

## ❖ 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.



```
ubuntu@ip-192-168-2-113:~$ sudo apt install mariadb-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package mariadb-server
ubuntu@ip-192-168-2-113:~$ sudo apt update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [8570 kB]
Get:6 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [1904 kB]
17% [5 Packages 490 kB/8570 kB 6%] [6 Packages 2685 B/1904 kB 0%]
```

### ➤ 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!

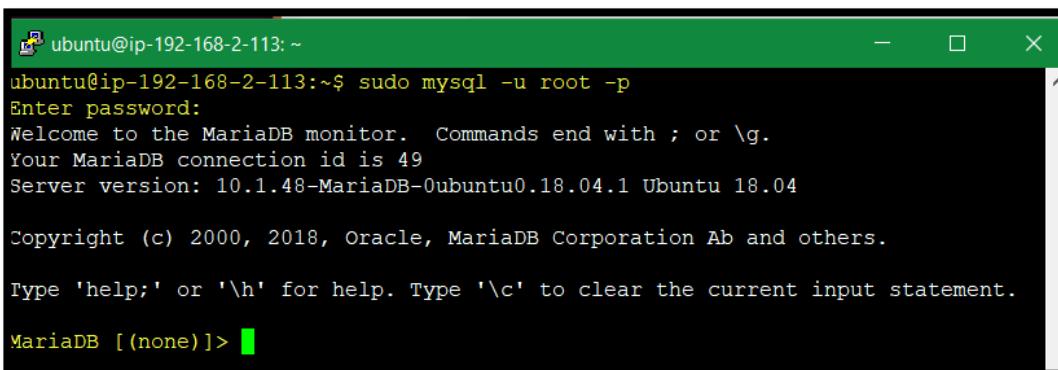
Cleaning up...

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.



A terminal window titled "ubuntu@ip-192-168-2-113: ~". The window shows the MySQL command-line interface. The user has entered "sudo mysql -u root -p" and is prompted for a password. After entering the password, the MySQL monitor welcome message is displayed, including the connection ID (49), server version (10.1.48-MariaDB-0ubuntu0.18.04.1 Ubuntu 18.04), and copyright information. The prompt "MariaDB [(none)]>" is visible at the bottom.

```
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.

```

ubuntu@ip-192-168-2-113:~$ ariaDB server version for the right syntax to use near ''Password@123'' at line 1
MariaDB [(none)]> exit
Bye
ubuntu@ip-192-168-2-113:~$ sudo mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 51
Server version: 10.1.48-MariaDB-Ubuntu0.18.04.1 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)]> create user 'owncloud'@'3.109.184.45' identified by 'Password@123';
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON *.* to owncloud@3.109.184.45 IDENTIFIED BY 'Password@123' WITH GRANT OPTION;
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON *.* to owncloud@'%' IDENTIFIED BY 'Password@123' WITH GRANT OPTION;
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]>

```

## ❖ Finalising ownCloud Installation:

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

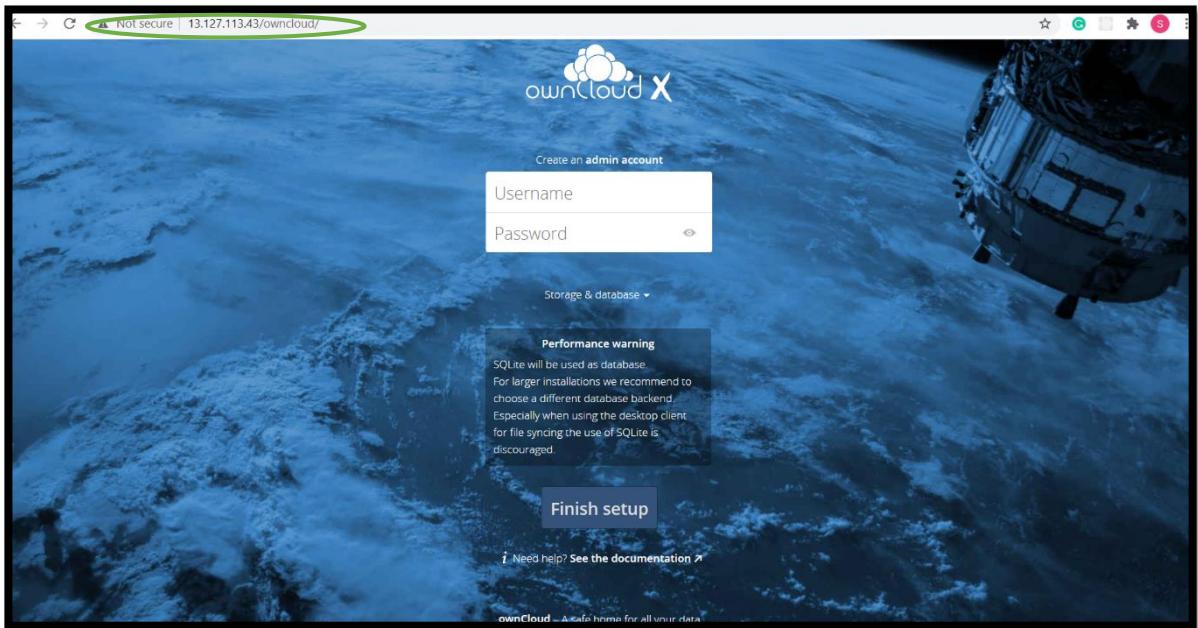
```
ubuntu@ip-192-168-2-113:~$ sudo systemctl restart mysql
ubuntu@ip-192-168-2-113:~$
```

```

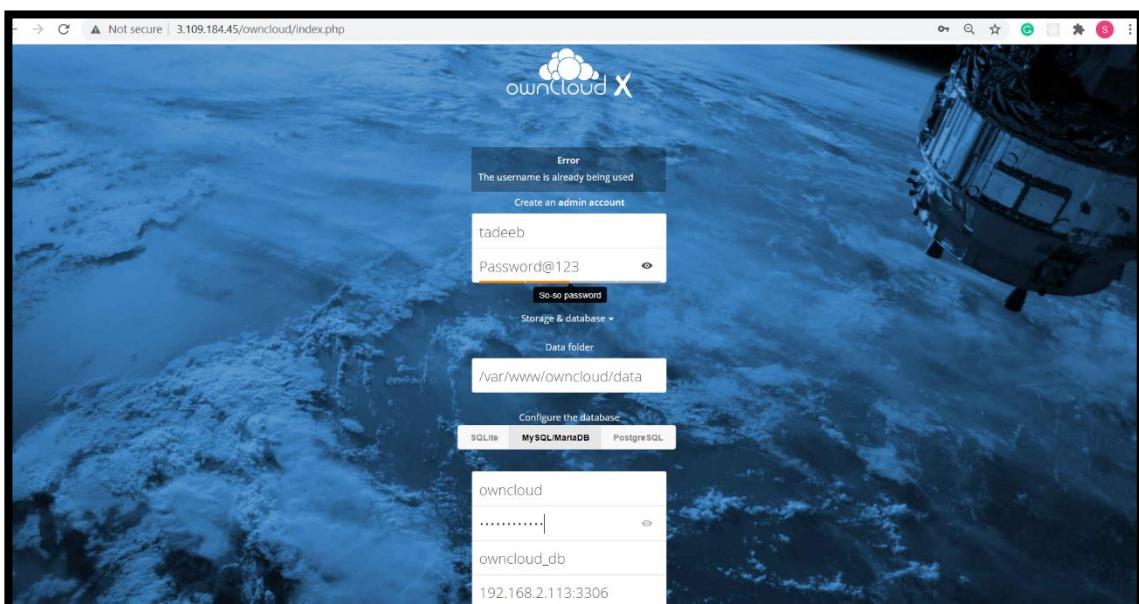
ubuntu@ip-192-168-1-213:~$ 
Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: 
Drop-In: /lib/systemd/system/apache2.service.d
          └─apache2-systemd.conf
 Active: active (running) since Fri 2021-10-08 05:53:45 UTC; 6s ago
Process: 7247 ExecStop=/usr/sbin/apachectl stop (code=exited, status=0/SUCCESS)
Process: 6950 ExecReload=/usr/sbin/apachectl graceful (code=exited, status=0/S
Process: 7252 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCE
Main PID: 7270 (apache2)
      Tasks: 6 (limit: 1140)
     CGroup: /system.slice/apache2.service
             ├─7270 /usr/sbin/apache2 -k start
             ├─7274 /usr/sbin/apache2 -k start
             ├─7276 /usr/sbin/apache2 -k start
             ├─7277 /usr/sbin/apache2 -k start
             ├─7279 /usr/sbin/apache2 -k start
             └─7281 /usr/sbin/apache2 -k start

Oct 08 05:53:45 ip-192-168-1-213 systemd[1]: Stopped The Apache HTTP Server.
Oct 08 05:53:45 ip-192-168-1-213 systemd[1]: Starting The Apache HTTP Server...
Oct 08 05:53:45 ip-192-168-1-213 apachectl[7252]: AH00558: apache2: Could not re
Oct 08 05:53:45 ip-192-168-1-213 systemd[1]: Started The Apache HTTP Server.
lines 1-22/22 (END)
ubuntu@ip-192-168-1-213:~$ sudo systemctl restart apache2
ubuntu@ip-192-168-1-213:~$
```

- **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.



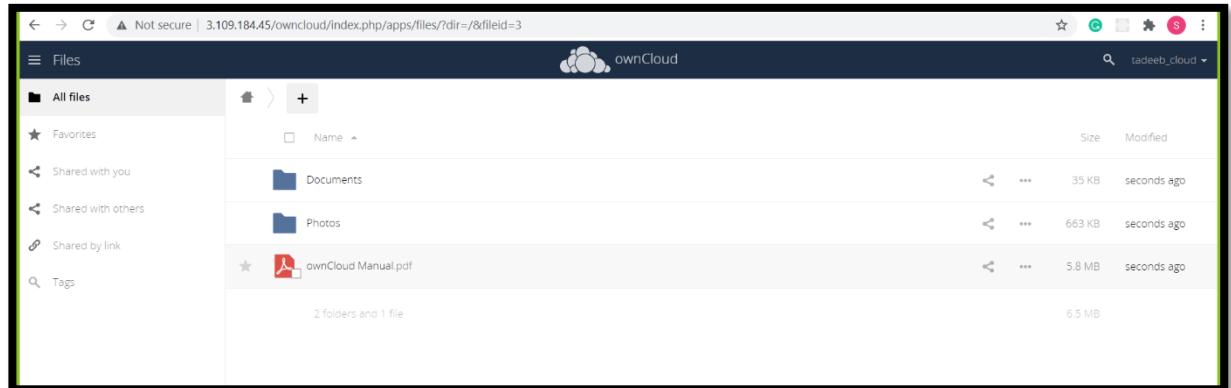
- **Step2)** Just below, click on 'Storage and database'. Select 'MySQL / MariaDB' under the 'configure the database' section and fill in the database credentials that you defined whilst creating the database for OwnCloud i.e database user, password of the database user, & database name. Note enter Private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



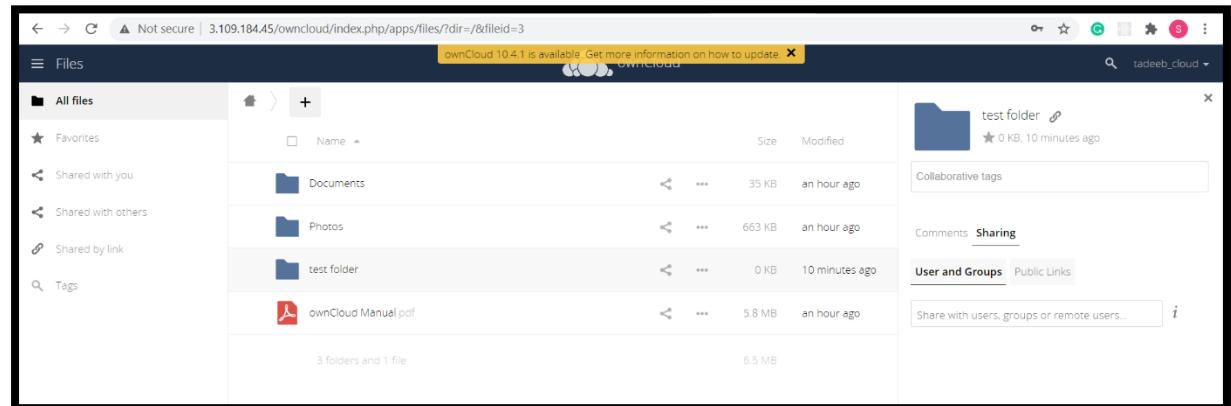
## ❖ Conclusion:

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.

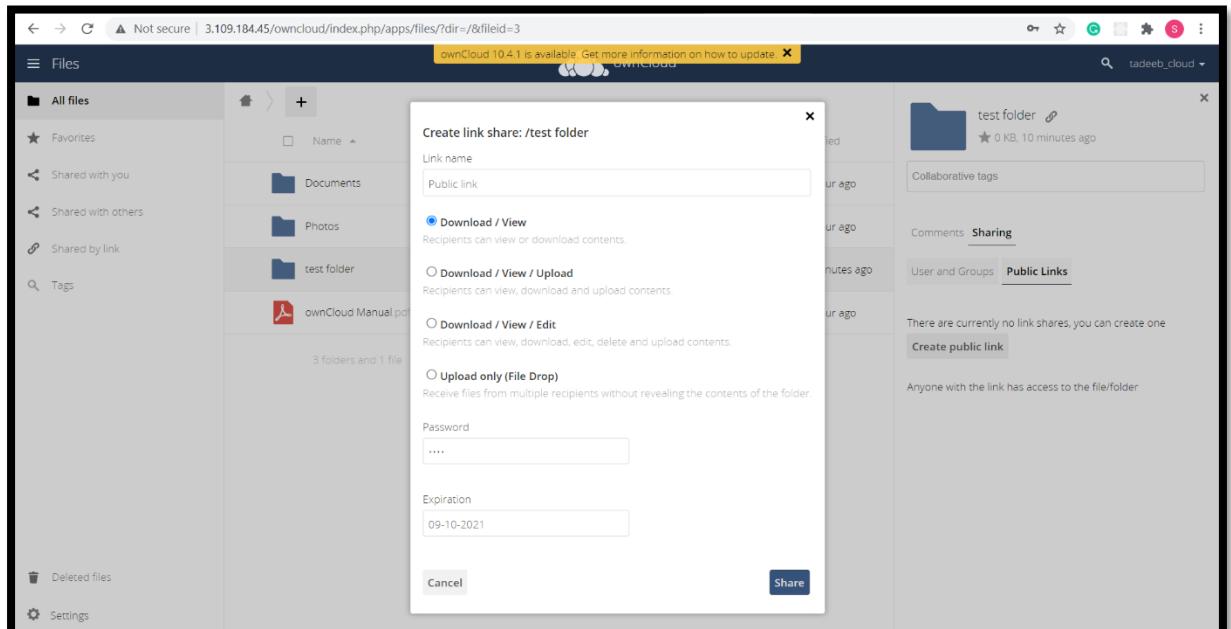
**Initially these are the default folders in my ownCloud:**



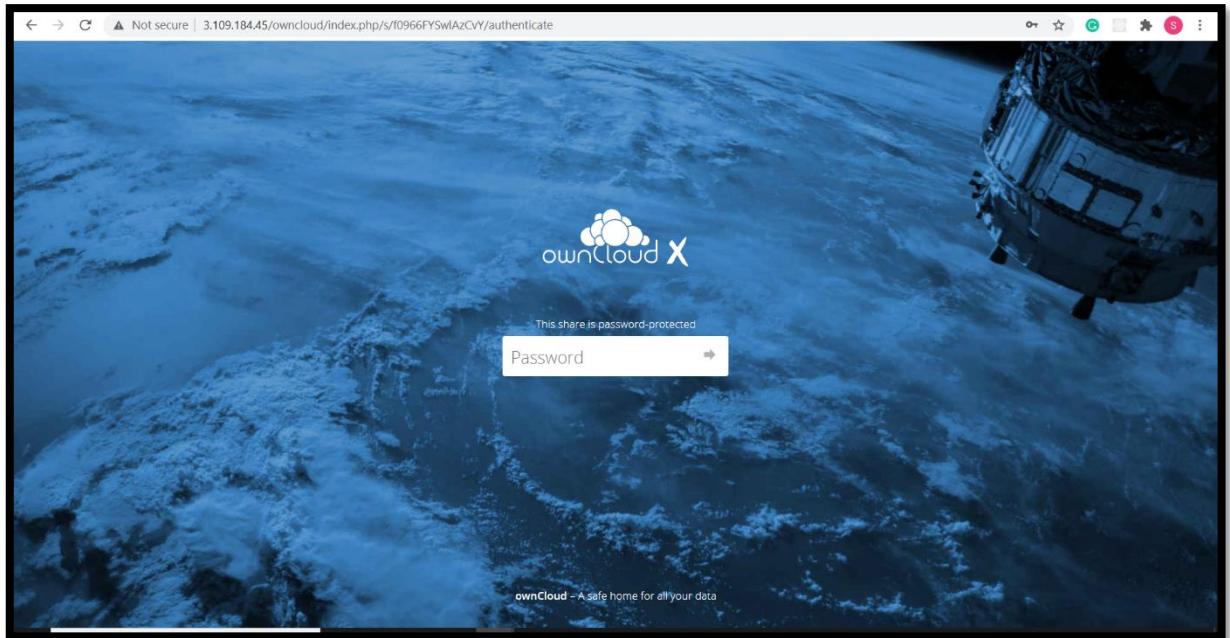
**I created a “test folder” for sharing with other members of my own enterprise.**



**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.