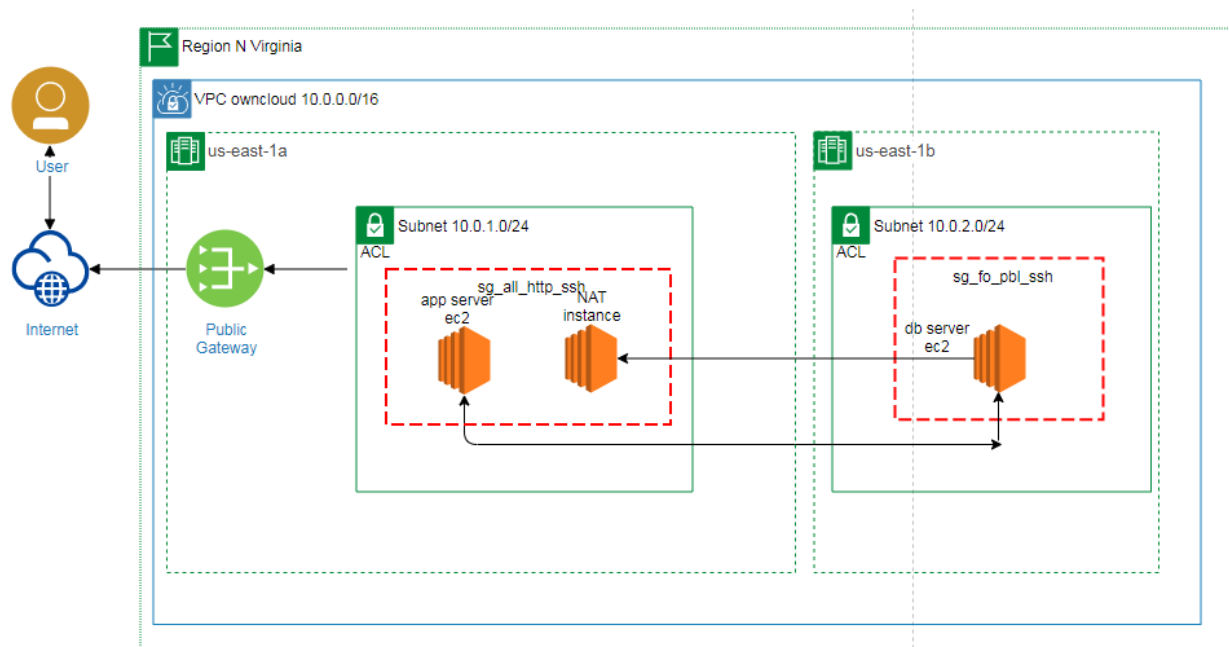


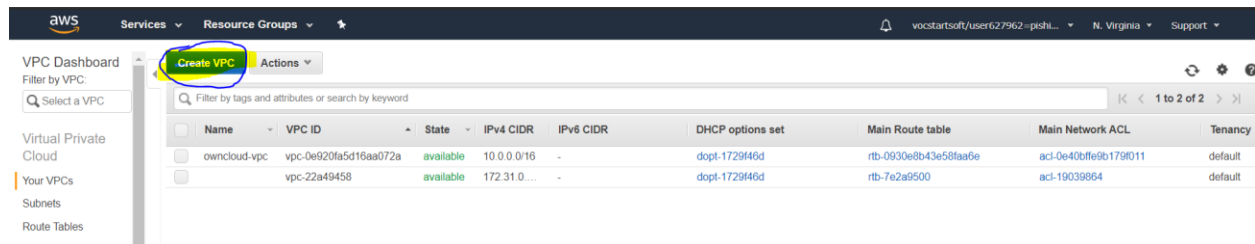
## 1. Phase 1 – Architecture:



## 2. Phase 2 – Implementation:

Creating the Custom VPC.

Step 1. Click on Create VPC | VPC Management Console



Step2:

- ✓ Enter the Name Tag (Owncloud-VPC)
- ✓ Enter the IPV4 CIDR Block (10.0.0.0/16)
- ✓ No need to change the IPv6 CIDR block and Tenancy
- ✓ Click on Create

## Create VPC

A VPC is an isolated portion of the AWS cloud populated by AWS objects, such as Amazon EC2 instances. You must specify an IPv4 address range for your VPC. Specify the IPv4 address range as a Classless Inter-Domain Routing (CIDR) block; for example, 10.0.0.0/16. You cannot specify an IPv4 CIDR block larger than /16. You can optionally associate an IPv6 CIDR block with the VPC.

Name tag

IPv4 CIDR block\*

IPv6 CIDR block ☐ No IPv6 CIDR Block ☐ Amazon provided IPv6 CIDR block ☐ IPv6 CIDR owned by me

Tenancy

\* Required

Cancel Create

## Step 3: Now we need to create a Public and Private Subnets

- ✓ Click on Create Subnet
- ✓ Enter Name tag (Public1)
- ✓ Click on the dropdown and select the custom VPC which you had created earlier.
- ✓ Select the availability zone. Right now, I am using the US-EAST-1 Region so that I can see the six availability zones. In this case, I am using (us-east-1a)
- ✓ Enter the CIDR range (10.0.1.0/24)

## Create subnet

Specify your subnet's IP address block in CIDR format; for example, 10.0.0.0/24. IPv4 block sizes must be between a /16 netmask and /28 netmask, and can be the same size as your VPC. An IPv6 CIDR block must be a /64 CIDR block.

Name tag

VPC\*

Availability Zone

VPC CIDRs	CIDR	Status	Status Reason
	10.0.0.0/16	associated	

IPv4 CIDR block\*

\* Required

Cancel Create

- ✓ Repeat the same process and create another public subnet in (us-east-1b) with CIDR 10.0.2.0/24
- ✓ Repeat the same process and create two private subnets one is in (us-east-1c with CIDR 10.0.3.0/24) another one is in (us-east-1d with CIDR 10.0.4.0/24)
- ✓ Once you created the four subnets, you will see the below screen with two public and two private subnets.

Create subnet

Actions

Filter by tags and attributes or search by keyword

<<

1 to 10 of 10

>>

<input type="checkbox"/>	Name	Subnet ID	State	VPC	IPv4 CIDR	Available IPv4	IPv6 CIDR	Availability Zone	Availability Zone ID	Route table	Network ACL
<input type="checkbox"/>	Private2	subnet-0114314ee5b0dba9a	available	vpc-0e920fa5d16aa072a	10.0.4.0/24	251	-	us-east-1d	use1-az2	rtb-08116aabe9aa87bcd   Priv...	acl-0e40bffe9b179011
<input type="checkbox"/>	Private1	subnet-02812b393eb8a914b	available	vpc-0e920fa5d16aa072a	10.0.3.0/24	251	-	us-east-1c	use1-az1	rtb-08116aabe9aa87bcd   Priv...	acl-0e40bffe9b179011
<input type="checkbox"/>	Public2	subnet-0749c2637ebb4933	available	vpc-0e920fa5d16aa072a	10.0.2.0/24	251	-	us-east-1b	use1-az6	rtb-0f0bf0e09b382f1b6d   Public...	acl-0e40bffe9b179011
<input type="checkbox"/>	Public1	subnet-9bcdeb143bc78eb1d	available	vpc-0e920fa5d16aa072a	10.0.1.0/24	251	-	us-east-1a	use1-az4	rtb-0f0bf0e09b382f1b6d   Public...	acl-0e40bffe9b179011

#### Step 4: Create an Internet gateway

- ✓ Select the Internet Gateways from VPC console
- ✓ Click on create **Create Internet Gateway**
- ✓ Enter the name of internet gateway

[Internet gateways](#) > Create internet gateway

#### Create internet gateway

An internet gateway is a virtual router that connects a VPC to the internet. To create a new internet gateway specify the name for the gateway below.

Name tag

\* Required

[Cancel](#) [Create](#)

- ✓ In my case, I entered the Owncloud-IGW. You will see the below screen after you create.

Create internet gateway					
Actions					
Filter by tags and attributes or search by keyword					
	Name	ID	State	VPC	Owner
<input checked="" type="checkbox"/>	owncloud-I...	igw-0bc862c57e6...	attached	vpc-0e920fa5d16...	174682964031
<input type="checkbox"/>		igw-228bec59	attached	vpc-22a49458	174682964031

- ✓ Now we need to attach the VPC to our IGW
- ✓ Select the IGW which you have created and click on actions and select the attach to VPC.
- ✓ Click on the dropdown button, and you will see our custom VPC, Select the VPC and click on attach.

Step 5: Now we need to create Route tables for your subnets (You can explicitly associate a subnet with a particular route table. Otherwise, the Subnet is implicitly associated with the main route table. A subnet can only be associated with one route table at a time, but you can associate multiple subnets with the same subnet route table.)

- ✓ Select the **Route tables** from left side VPC panel
- ✓ Click on **create route table**
- ✓ Enter the name tag **public Route** and select our custom VPC from the dropdown.

- ✓ Repeat the same steps and create one more route table with the **private route** name tag and select our custom VPC from the dropdown.

[Route Tables](#) > Create route table

## Create route table

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

Name tag

VPC\*

\* Required

[Cancel](#) [Create](#)

- ✓ Once you created the two route tables, you will see the below screen.

[Create route table](#) [Actions](#)

Filter by tags and attributes or search by keyword

<input type="checkbox"/>	Name	Route Table ID	Explicit subnet association	Edge associations	Main	VPC ID	Owner
<input type="checkbox"/>		rtb-0930e8b43e58faa6e	-	-	Yes	vpc-0e920fa5d16aa072a  ...	174682964031
<input type="checkbox"/>		rtb-7e2a9500	-	-	Yes	vpc-22a49458	174682964031
<input type="checkbox"/>	PrivateRoute	rtb-08116aabe9aa87bcd	2 subnets	-	No	vpc-0e920fa5d16aa072a  ...	174682964031
<input checked="" type="checkbox"/>	PublicRoute	rtb-0f8bfe09b382f1b6d	2 subnets	-	No	vpc-0e920fa5d16aa072a  ...	174682964031

- ✓ Select the publicRoute and click on routes → edit routes
- ✓ Attach IGW to this route table so that subnets which are associated with this route table they will get the internet.

[Route Tables](#) > Edit routes

## Edit routes

Destination	Target	Status	Propagated
10.0.0.0/16	local	active	No
0.0.0.0/0	igw-0bc862c57e67a355d	active	No

[Add route](#)

igw-0bc862c57e67a355d owncloud-IGW

\* Required

[Cancel](#) [Save routes](#)

- ✓ Select the subnet associations → click on Edit Subnet associations → select the public subnets → click on save

## Edit subnet associations

Route table rtb-0f8bfe09b382f1b6d (PublicRoute)

Associated subnets subnet-0749c263f7ebb4933 subnet-0bcdeb143bc78eb1d

Filter by attributes or search by keyword				
K < 1 to 4 of 4 > X				
<input type="checkbox"/>	Subnet ID	IPV4 CIDR	IPV6 CIDR	Current Route Table
<input type="checkbox"/>	subnet-0114314ee5b0dba9a   Private2	10.0.4.0/24	-	rtb-08116aabe9aa87bcd
<input checked="" type="checkbox"/>	subnet-0bcdeb143bc78eb1d   Public1	10.0.1.0/24	-	rtb-0f8bfe09b382f1b6d
<input type="checkbox"/>	subnet-02812b393eb8a914b   Private1	10.0.3.0/24	-	rtb-08116aabe9aa87bcd
<input checked="" type="checkbox"/>	subnet-0749c263f7ebb4933   Public2	10.0.2.0/24	-	rtb-0f8bfe09b382f1b6d

\* Required

Cancel Save

- ✓ Respect the same step for the Private Route
- ✓ Select the private route → Select the subnet associations → click on Edit Subnet associations → select the Private subnets → click on save
- ✓ But in this case, we are not adding the IGW in the private route table. So, the traffic will not leave the VPC.

## Creating the NAT instance.

- ✓ Click on service → Select the ec2 from compute category.
- ✓ Select the launch instance → under the Operating system select the amazon Linux → Enter the nat in the search box → press enter
- ✓ You will get below screen

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: nat

Category	AMI ID	AMI Name	Root device type	Virtualization type	ENI Enabled	Architecture	Buttons
Quick Start (0)							
My AMIs (0)							
AWS Marketplace (22)							
Community AMIs (46)							
Operating system							
Amazon Linux							
	ami-00a9d4a05375b2763	Amazon Linux AMI 2018.03.0.20181116-x86_64 VPC HVM EBS	ebd	hvm	Yes	64-bit (x86)	Select
	ami-01523d7b	Amazon Linux AMI 2017.09.1.20180108-x86_64 VPC NAT HVM EBS	ebd	hvm	Yes	64-bit (x86)	Select

- ✓ Select **ami-00a9d4a05375b2763** → click on select
- ✓ Choose the instance as t2.micro → click on next
- ✓ In the networking, the section clicks on the dropdown → select the custom VPC which we have created earlier
- ✓ Subnet section → select the public Subnet → select enable in auto assign Public IP

**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 [Launch into Auto Scaling Group](#)

Purchasing option: ☐ Request Spot instances

Network: vpc-0e920fa5d16aa072a | owndcloud-vpc [Create new VPC](#)

Subnet: subnet-0bcdeb143bc78eb1d | Public1 | us-east-1a [Create new subnet](#)  
249 IP Addresses available

Auto-assign Public IP: Enable

Placement group: ☐ Add instance to placement group

Capacity Reservation: Open [Create new Capacity Reservation](#)

IAM role: None [Create new IAM role](#)

Shutdown behavior: Stop

Stop - Hibernate behavior: ☐ Enable hibernation as an additional stop behavior

Enable termination protection: ☐ Protect against accidental termination

Monitoring: ☐ Enable CloudWatch detailed monitoring  
[Additional charges apply](#)

Tenancy: Shared - Run a shared hardware instance  
[Additional charges will apply for dedicated tenancy](#)

Elastic Inference: ☐ Add an Elastic Inference accelerator  
[Additional charges apply](#)

T2/T3 Unlimited: ☐ Enable  
[Additional charges may apply](#)

File systems: [Add file systems](#) [Create new file system](#)

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Add Storage](#)

- ✓ Leave as it is remaining fields and click on Next: add storage
- ✓ For our demo purpose, we do not add volume; we are keeping as it is → now click on next add: Tags → add the tags → add security groups
- ✓ For our demo, I am allowing the only Http, and Https port's only for IPV4

**Inbound rules** | Outbound rules | Tags

[Edit inbound rules](#)

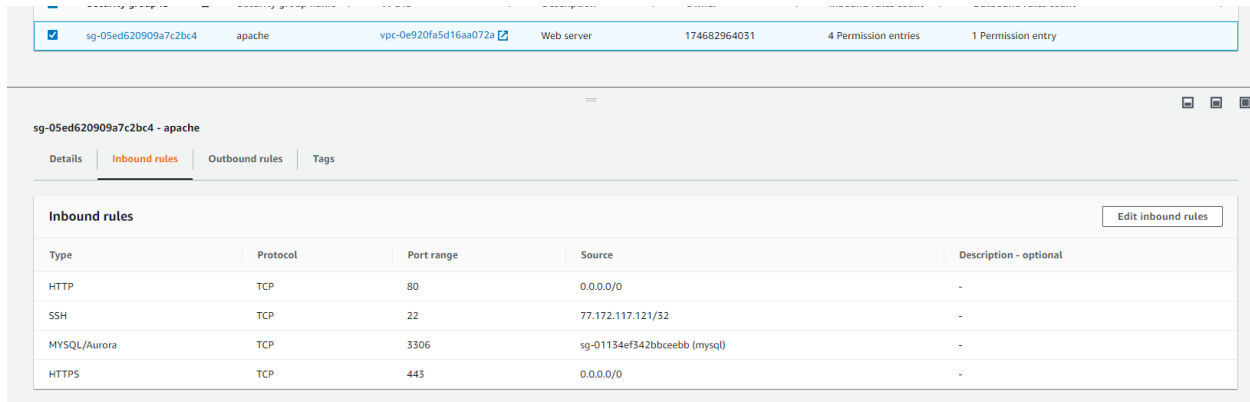
Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	0.0.0.0/0	-
HTTPS	TCP	443	0.0.0.0/0	-

- ✓ Click on Review and Launch

## Creating Ec2 instance in Public Subnet and installation of apache and PHP.

- ✓ Go to ec2 console
- ✓ Click on launch instance → select the **Ubuntu Server 18.04 LTS (HVM), SSD Volume Type** - ami-07ebfd5b3428b6f4d → Chose the instance type t2.micro → click on Next: Configure Instance Details
- ✓ Select our custom VPC from the dropdown in VPC section
- ✓ Select the Public Subnet from the dropdown in the Subnet section
- ✓ Click on add storage
- ✓ No need to change the storage section for our demo → click on add: Tags
- ✓ Add the tags (Key → Name & Value: Apache) → Click on configure security groups
- ✓ Allow Http and Https from anywhere and ssh from your computer
- ✓ Once you are done with the creation of MySQL server come back to a security group and allow MySQL from that server specifically

✓ Please find the below screenshot for your reference.



The screenshot shows the AWS IAM console interface for a security group named 'sg-05ed620909a7c2bc4 - apache'. The 'Inbound rules' tab is selected, displaying a table of inbound rules. The table has columns for Type, Protocol, Port range, Source, and Description - optional. There are four rules listed: HTTP (TCP, port 80, source 0.0.0.0/0), SSH (TCP, port 22, source 77.172.117.121/32), MySQL/Aurora (TCP, port 3306, source sg-01134ef342bbceebb (mysql)), and HTTPS (TCP, port 443, source 0.0.0.0/0). An 'Edit inbound rules' button is visible in the top right corner of the table.

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	0.0.0.0/0	-
SSH	TCP	22	77.172.117.121/32	-
MySQL/Aurora	TCP	3306	sg-01134ef342bbceebb (mysql)	-
HTTPS	TCP	443	0.0.0.0/0	-

- ✓ Download the .pem file and Click on review and launch
  - ✓ Login to the server using .pem file
  - ✓ Run the below commands to install required packages for owncloud
- ```
# sudo apt-get update
# sudo apt-get install apache
# sudo apt install php libapache2-mod-php php-mysql
# vim /etc/apache2/mods-enabled/dir.conf
✓ update the configuration file
```

```
<IfModule mod_dir.c>
    DirectoryIndex index.php index.html index.cgi index.pl index.php index.xhtml index.htm
</IfModule>

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

- ```
# sudo systemctl restart apache2
# curl https://attic.owncloud.org/download/repositories/10.0/Ubuntu_18.04/Release.key
| sudo apt-key add -
# sudo apt update
# apt install php-bz2 php-curl php-gd php-imagick php-intl php-mbstring php-xml php-zip
owncloud-files
# vim /etc/apache2/sites-enabled/000-default.conf
```

```

<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com

    ServerAdmin webmaster@localhost
    #DocumentRoot /var/www/html
    DocumentRoot /var/www/owncloud

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

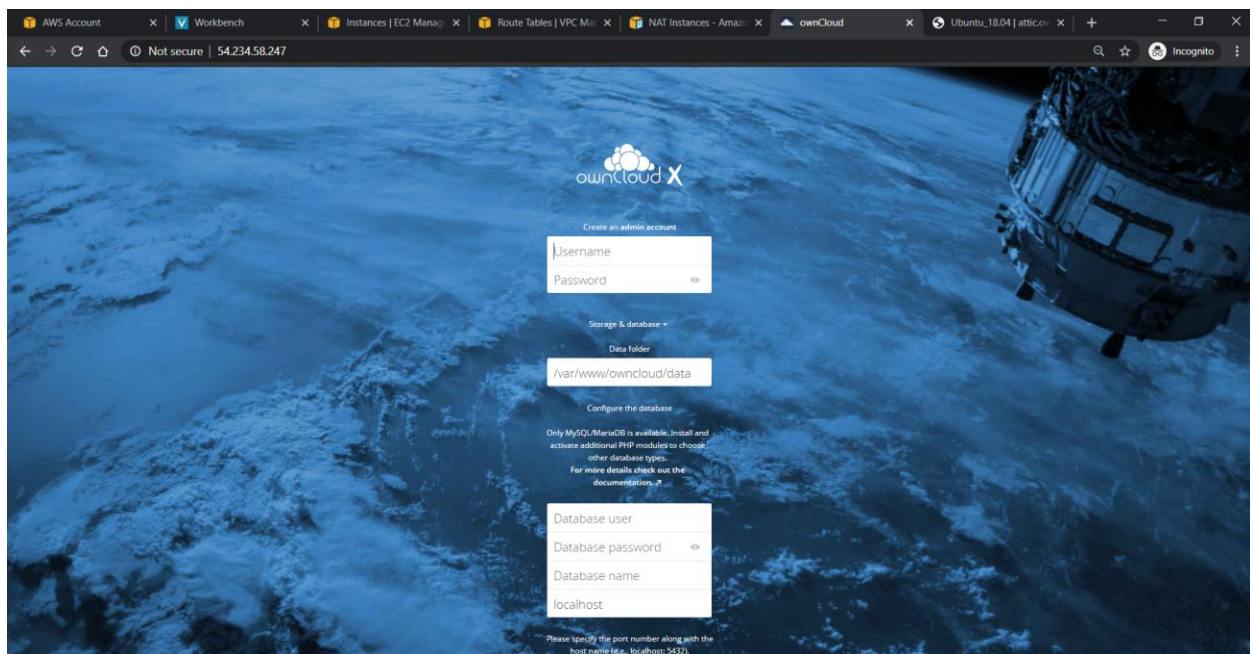
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    # For most configuration files from conf-available/, which are
    # enabled or disabled at a global level, it is possible to
    # include a line for only one particular virtual host. For example the
    # following line enables the CGI configuration for this host only
    # after it has been globally disabled with "a2disconf".
    #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

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

```

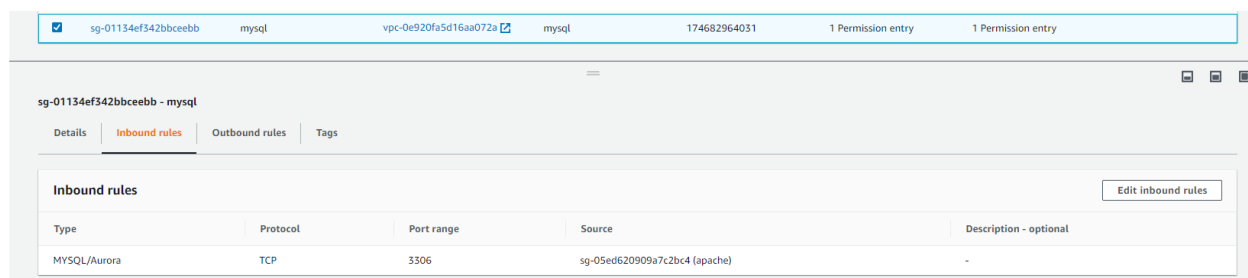
- # sudo systemctl reload apache2
- ✓ Go to the browser and enter the IP address of your server
- ✓ You will see the below screen



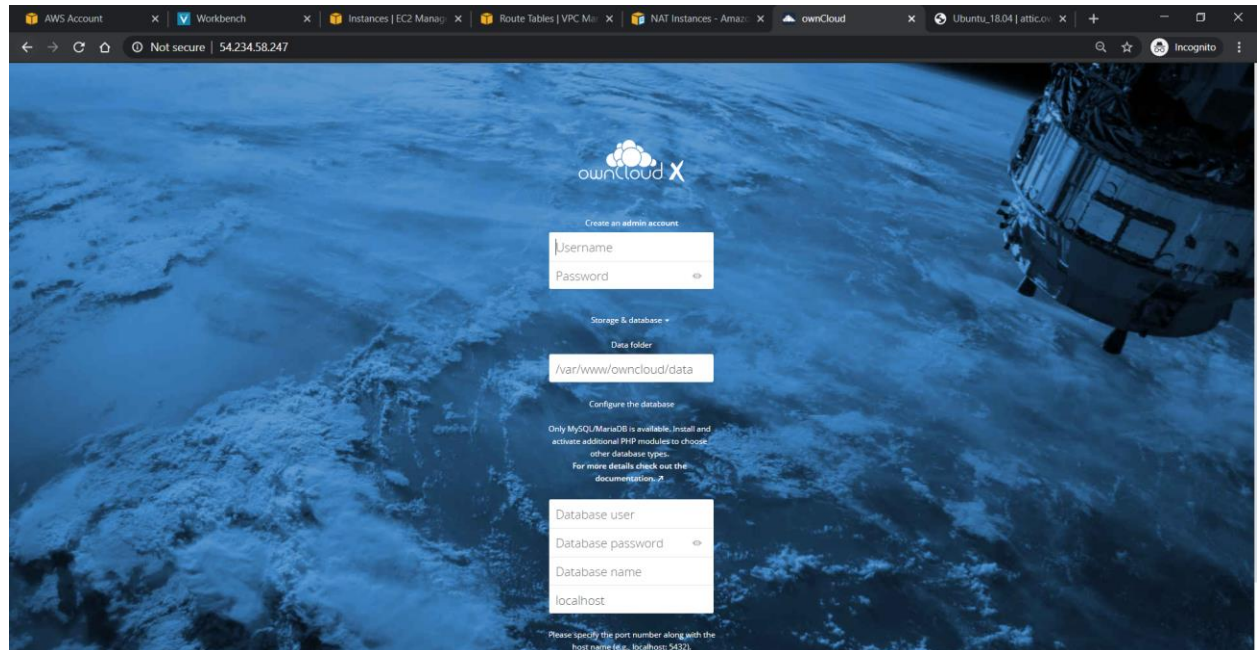


## Creating Ec2 instance and installation of MySQL in private Subnet.

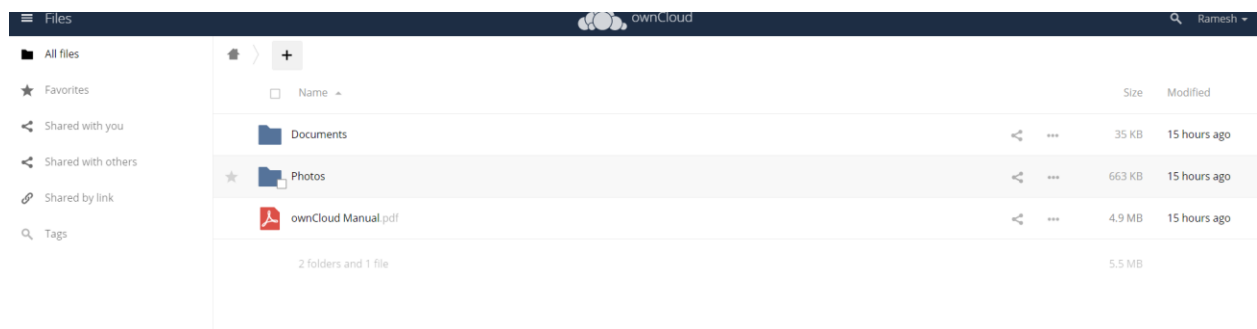
- ✓ Go to ec2 console
- ✓ Click on launch instance → select the **Amazon Linux 2 AMI (HVM), SSD Volume Type** → Chose the instance type t2.micro → click on Next: Configure Instance Details
- ✓ Select our custom VPC from the dropdown in VPC section
- ✓ Select the private Subnet from the dropdown in the Subnet section
- ✓ Click on add storage
- ✓ No need to change the storage section for our demo → click on add: Tags
- ✓ Add the tags (Key → Name & Value: Mysql) → Click on configure security groups
- ✓ Please find the below screenshot for your reference.



- ✓ Now click on review and launch
  - ✓ To install the MySQL servers and to create and configure a database you need to login to server
  - ✓ For the temporary purpose, I am allowing the 22, 80, 443 ports from the webserver.
  - ✓ Copy the .pem file from your local machine to webserver to connect to a server
  - ✓ Use ssh -I <pem file> username@<server IP> command to log in the server
  - ✓ Run the below command to install and configure the database.
- ```
# wget http://repo.mysql.com/mysql-community-release-el7-5.noarch.rpm
# sudo rpm -ivh mysql-community-release-el7-5.noarch.rpm
# yum update
# sudo yum install mysql-server
# sudo systemctl start mysqld
```
- ✓
  - ✓ Create a database
  - ✓ Create a user and grant permission to access the database
  - ✓ Now go to the webserver and try to access the page you will see the below screen.



- ✓ Fill the details and click on finish.
- ✓ After successful installation, you will see the below screen.



## Lessons and observations

- ✓ Learnt the purpose VPC, Public and Private Subnet, Route tables, Nat instance, Security groups and traffic between the servers.
- ✓ Route tables are the ones that allow you to connect to instances or external sources at the subnet level, whereas the security groups work at the instance level.
- ✓ I observed that the default VPC always connected to the internet. So all the subnets in the default VPC are public by default.