CONTAINER

Created By:- vinay chawan

Project Objectives and Requirements

Prelude-From Infrastructure to Containers

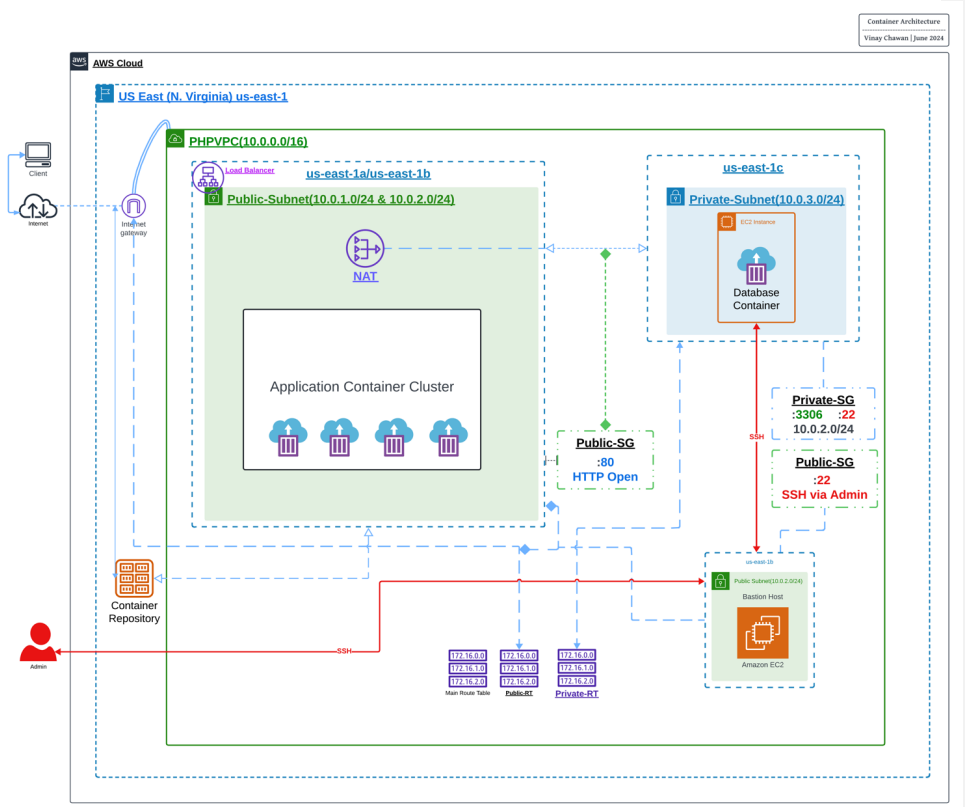
Containerization of an application has a distinct advantage over an equivalent IaaS deployment, in that the deployment becomes essentially hardware architecture-independent and can easily be ported to multiple platforms with minimal effort. This is a stark contrast to infrastructure-based deployments where multiple factors must be considered, such as operating system, hardware, underlying libraries and dependencies etc. According to Gartner, the container management market has seen accelerated growth over the past year. The market is forecast to reach $1.4 billion by 2025, with a 25.1% CAGR. As per their estimate, by the end of 2022, more than 75% of global organizations will be running containerized applications in production, up from less than 30% in 2020.As a result of this, enterprise demand for container management is increasing.

Container management provides software and/or services that support the management of containers, at scale, in production environments. In the last 2 decades small and large enterprises have invested heavily in developing bespoke applications. Since these applications have been built and enhanced over a period, they are complex and any form of reengineering to convert it to a smaller modularized independently hosted services is difficult. With the advent of cloud and containerization, these organizations are looking to take advantage of the predictable packaging of these “medium sized” applications and leverage the managed container services from the cloud. The objective of this project is to experience such a scenario and move a classical (simplified) web application to the cloud. You need to deploy the PHP application on an ECS cluster, and the backend database being hosted on an EC2 instance running a MySQL container on Docker.

The database container will need to be exposed for remote connections and logins. The PHP application should be configured with the database parameters and credentials and have the required libraries enabled in the container.

Created By: Vinay Chawan

Architecture Diagram



Services Used

- **Networking Services:**
  - **VPC (Virtual Private Cloud)**
  - **Subnets**
  - **Route Table**
  - **Internet Gateway (IGW)**
  - **NAT Gateway**

- **Compute Services:**
  - **EC2(Elastic Compute Cloud)**
  - **Security Group**
  - **Amazon Linux 2(Free Tier)**
  - **Load Balancer**
  - **Target Group**

- **Storage & IAM:**
  - **S3 Bucket**
  - **ECR (Elastic Container Registry)**
  - **IAM (Identity Access Management) (Roles)**
  - **ECS (Elastic Container Service)**

- **Security Considerations**:
  - VPC (10.0.0.0/16)
  - VPC (10.0.0.0/16)
  - Database (Private IP)
  - IGW (Internet Gate for internet access) attachment on VPC for open world access to web application. Also, for update/package installations on Database server.
  - Open HTTP to ECS for Web Applications access.
  - NAT for Setup/Update of packaged on Database server via Public Subnet.
  - MySQL connect between Public and Private via port 3306.
  - SSH connection to Database server via Public (Bastion Host)

Network Configuration/Setup

- VPC (Virtual Private Cloud):
  - Name: PHPVPC
- Subnet:
  - Public 1. Public (10.0.1.0/24) (us-east-1a)(Enabled Public IP assignment)
  - Public (10.0.2.0/24) ()(Only Private IP)(us-east-1b)
  - Private (10.0.3.0/24) ()(Only Private IP)(us-east-1c)

- Route Table:
  - Create 2 “Public” route table.
  - Create “Private” route table.

- Internet Gateway:
  - Create Internet gateway (IGW)

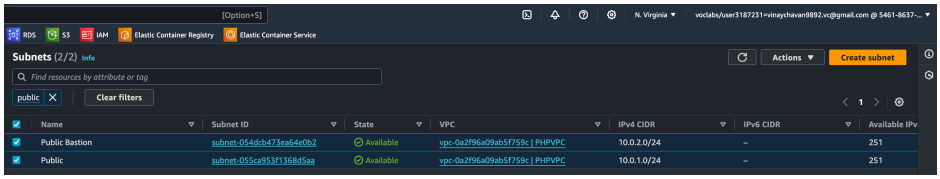
- NAT Gateway:
  - Create NAT gateway (NAT)

Installation and configuration process

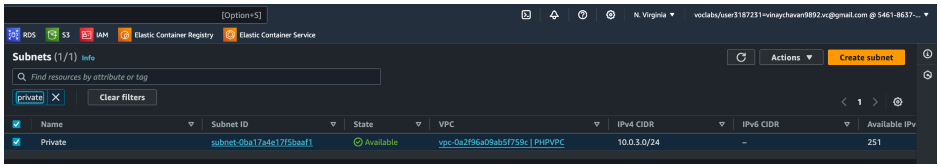
- Network Setup/Configuration

  - VPC:
    - Create Virtual Private Cloud (VPC) Create Virtual Private Cloud(VPC) as PHPvpc.
    - Assign IPv4 CIDR as 10.0.0.0/16
    - Keep rest of the setting Default.
    - Add the Name tag with PHPvpc.
    - Click on “Create VPC”.
    -
  - Subnet:
    - Create 3 subnets.(Public, Public and Private)
    - Click on “Create Subnet”
    - Select the VPC(PHPVPC)
    - First subnet name is Public with AZ is us-east-1a and IPv4 is 10.0.1.0/24.
    - Tag Name as Public.
    - Click on “Add new Subnet”.
    - Second subnet name is Public for bastion with AZ is us-east-1b and IPv4 is 10.0.2.0/24
    - Tag Name as Public.
    - Third subnet name is Private with AZ is us-east-1c and IPv4 is 10.0.3.0/24
    - Tag Name as Public.
    - Click on “Create Subnet”.
    - Tick the Public Subnet.
    - Open “Actions”, Click on “Edit Subnet Settings”.
    - Tick on “Enable Auto Assign public IPv4 Address”.
    - Save

Public Subnet:

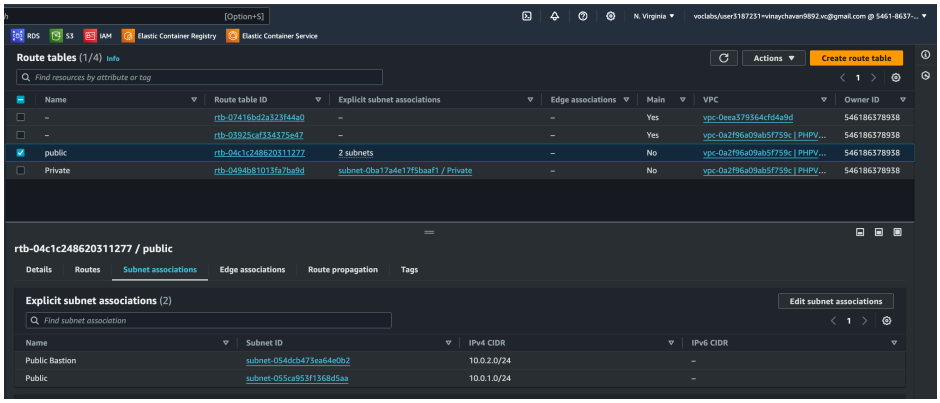


Private Subnet:

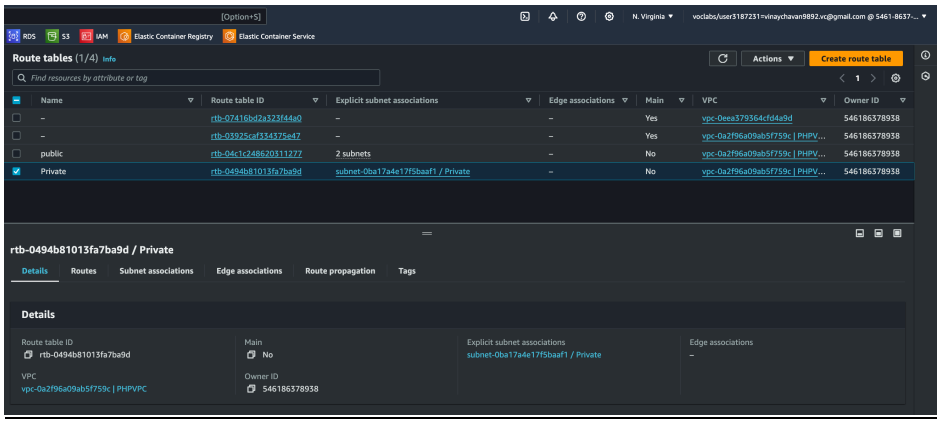


- ❖ Route Table:
  - ➢ Create 3 Route tables.
  - ➢ Click on “Create Table”
  - ➢ Set table name, Select the PHPVPC.
  - ➢ Tag as per requirement.

Public Route Table:



Private Route Table:



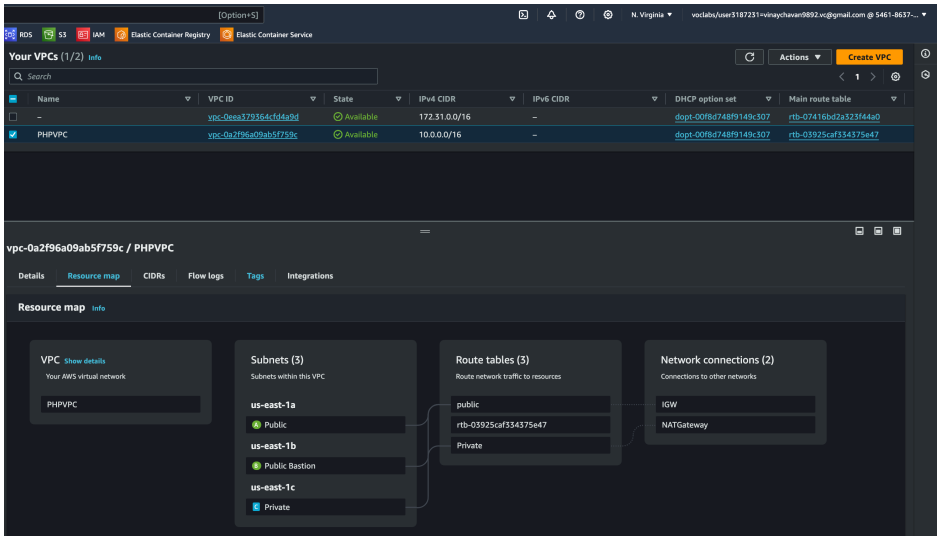
- **Internet Gateways (IGW)**
  - Click on Internet Gateways.
  - Click on “Internet gateway”.
  - Name and tag the internet gateway as per requirement.
  - Click on Create Internet gateway.
  - Click on “Attach to a VPC”
  - Select the OwnCloud-VPC.
  - Click on Attach internet gateway.
  - Click on Route tables.
  - Select “Public” route table.
  - Click on “Routes” tab.
  - Click on Edit routes.
  - Click on Add route.
  - Select 0.0.0.0/0, Select “Internet gateway” and Select “IGW-ID”
  - Then Save changes.

❖ NAT Gateways

- Click on NAT Gateways.
- Click on Create NAT gateway.
- Provide a name to NAT gateway.
- Select “Public” Subnet.
- Connective Type is Public.
- Click on Allocate Elastic IP.
- Click On “Create NAT Gateway”.
- Then go to Route table.
- Select “Private” route table.
- Click on “Routes” tab.
- Click on Edit routes.
- Click on Add route.
- Select 0.0.0.0/0, Select “NAT gateway” and select “NAT for Database”.
- Then Save changes.

Commented [vc1]: Page 17

Owncloud-VPC Resource map:



Compute and Instances

❖ Bastion Host:

- **Server Name: - Bastion host**
- **Subnet: - Public - 10.0.1.0/24**
- **AZ: -** us-east-1a
- Need IGW connection.
- **Security Group as follow: - SSH via Own IP**
- **For connecting to Database Container**

❖ Database Container:

- **Server name: - Database Container**
- **Subnet: - Private - 10.0.3.0/24**
- **AZ: -** us-east-1c
- **Needs NAT package installation**
- **Security Group as follow: - SSH via bastion host and Mysql via SG**

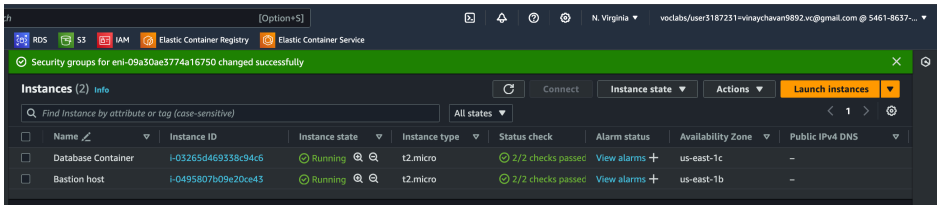
EC2(Elastic Compute Cloud)

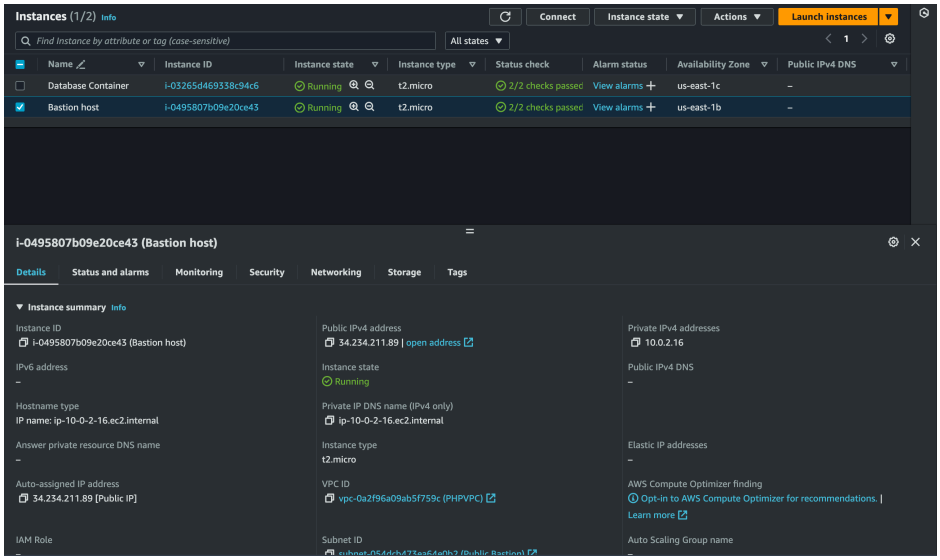
❖ Bastion Host

- First one is “Bastion host”.
- Click on Launch instance in EC2.
- Name first instance,
- Select AMI as “Amazon Linux 2”
- Create a Key Pair.
- In Network Settings, select “PHPVPC”.
- Subnet as Public.
- Confirm the “Auto assign Public IP” is Enabled.
- Create a security group of “SSH via my IP”.
- SSH :22 Source type: My IP.
- Rest settings as default.
- Click on Launch Instance.

❖ Database Container

- Second is “Database Container”.
- Click on Launch instance in EC2.
- Name the instance.
- Select AMI as “Amazon Linux 2.
- Create a Key Pair.
- In Network Settings, select “PHPVPC”.
- Subnet as Private.
- Confirm the “Auto assign Public IP” is Disabled.
- Create a security group of “SSH via bastion host and MYSql connect to SG group for ECS”.
- SSH :22 Source:10.0.2.0/24.
- MySQL/Aurora :3306 Source:
- Rest settings as default.
- Click on Launch Instance.


[Option+S]
N. Virginia
RDS
S3
IAM
Elastic Container Registry
Elastic Container Service
Security groups for eni-09a30ae3774a16750 changed successfully
Instances (2) Info
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4 DNS
Database Container
i-03265d469338c94c6
Running
t2.micro
2/2 checks passed
View alarms
us-east-1c
Bastion host
i-0495807b09e20ce43
Running
t2.micro
2/2 checks passed
View alarms
us-east-1b


Instances (1/2) Info
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4 DNS
Database Container
i-03265d469338c94c6
Running
t2.micro
2/2 checks passed
View alarms
us-east-1c
Bastion host
i-0495807b09e20ce43
Running
t2.micro
2/2 checks passed
View alarms
us-east-1b
i-0495807b09e20ce43 (Bastion host)
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags
Instance summary Info
Instance ID
i-0495807b09e20ce43 (Bastion host)
Public IPv4 address
34.234.211.89 | open address
Private IPv4 addresses
10.0.2.16
IPv6 address
–
Instance state
Running
Public IPv4 DNS
–
Hostname type
IP name: ip-10-0-2-16.ec2.internal
Private IP DNS name (IPv4 only)
ip-10-0-2-16.ec2.internal
Answer private resource DNS name
–
Instance type
t2.micro
Elastic IP addresses
–
Auto-assigned IP address
34.234.211.89 [Public IP]
VPC ID
vpc-0a2f96a09ab5f759c (PHPVPC)
AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations.
Learn more
IAM Role
Subnet ID
Auto Scaling Group name

Setup/Configuration of Database Container,ECR and ECS

Note: Perform below steps before setup/configuration of Servers.

(Database Container)

- Instance ID:- i-03265d469338c94c6
- (Database Container)
- Open an SSH client.
- Locate your private key file. The key used to launch this instance is DBContainer.pem
- Run this command, if necessary, to ensure your key is not publicly viewable.
- chmod 400 "DBContainer.pem"
- Connect to your instance using its Private IP:
- 10.0.3.17
- Example:
- ssh -i "DBContainer.pem" ec2-user@10.0.3.17

(Bastion Host)

Instance ID:- i-0495807b09e20ce43(Bastion host)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is Bastion host.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   chmod 400 "Bastion host.pem"
4. Connect to your instance using its Public IP:
   34.234.211.89
   Example:
   ssh -i "Bastion host.pem" ec2-user@34.234.211.89

Point's to be Noted:

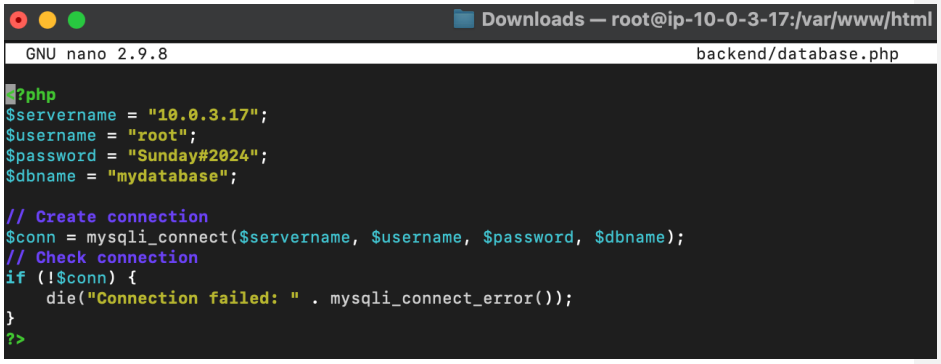
- Since the Database is in a Private Network by using Bastion host ,will need to take SSH.
- Hence need to send .pem to Database.
- Run this command, to send the Database.pem to Database Container.

scp -i bastion.pem ./datsbase.pem ec2-user@10.0.3.17:/home/ec2-user/Database.pem
scp -i DBContainer.pem crud.zip ec2-user@10.0.3.17:/home/ec2-user/

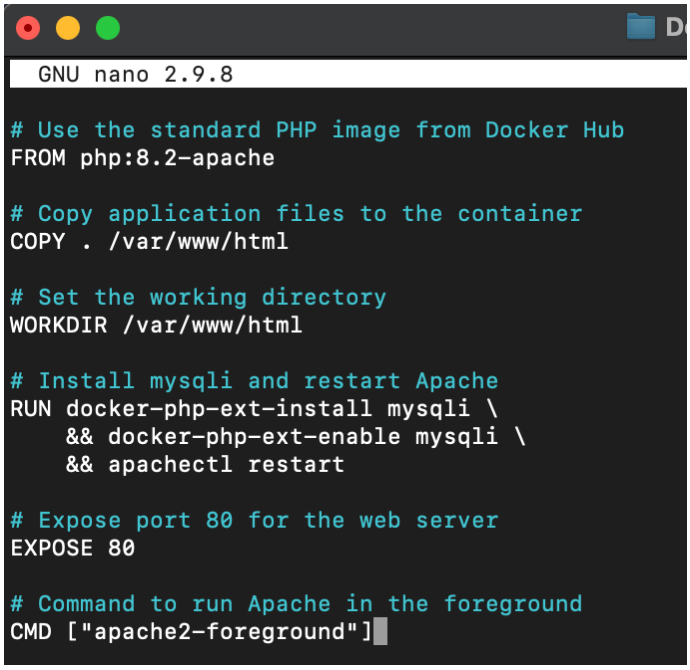
❖ **Database Container**

- sudo su
- yum update
- yum install docker* httpd* -y
- amazon-linux-extras install docker
- systemctl start docker
- systemctl start httpd
- usermod -aG docker ec2-user
- Relogin to shell to take effect.
- docker version
- docker images
- docker ps -a
- mv crud.zip /var/www/html/
- cd var/www/html/
- unzip crud.zip
- docker pull mysql
- docker pull php
- docker images
- docker network create mynetwork
- docker run --name mysql-container --network mynetwork -e MYSQL_ROOT_PASSWORD=Sunday#2024 -e MYSQL_DATABASE=mydatabase -p 3306:3306 -d mysql:latest
- docker cp /var/www/html/crud.sql mysql-container:/crud.sql
- docker exec -it mysql-container bash
- mysql -u root -p mydatabase < /crud.sql

- exit
- nano backend/database.php



- nano Dockerfile

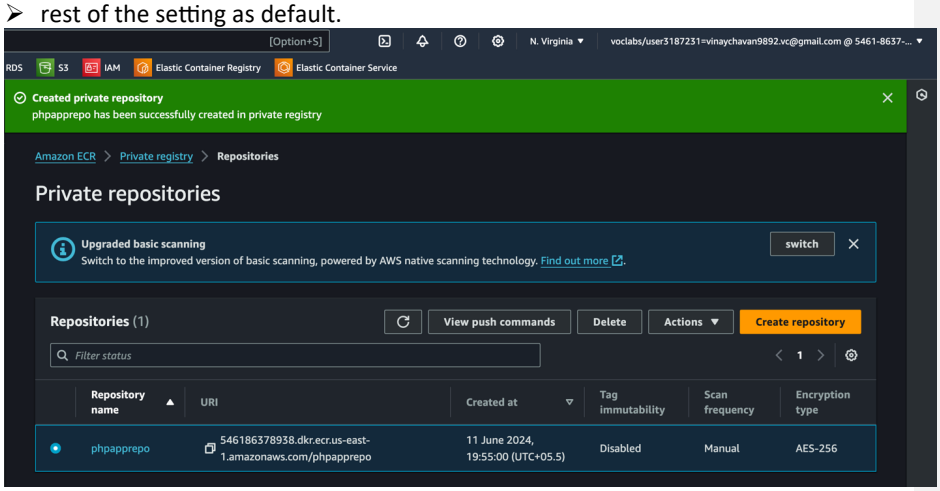


- docker build -t my-php-app .
- docker run --name php-container --network mynetwork -p 80:80 -d my-php-app
- yum install awscli
- aws configure
- AWS Access Key ID [None]: ha

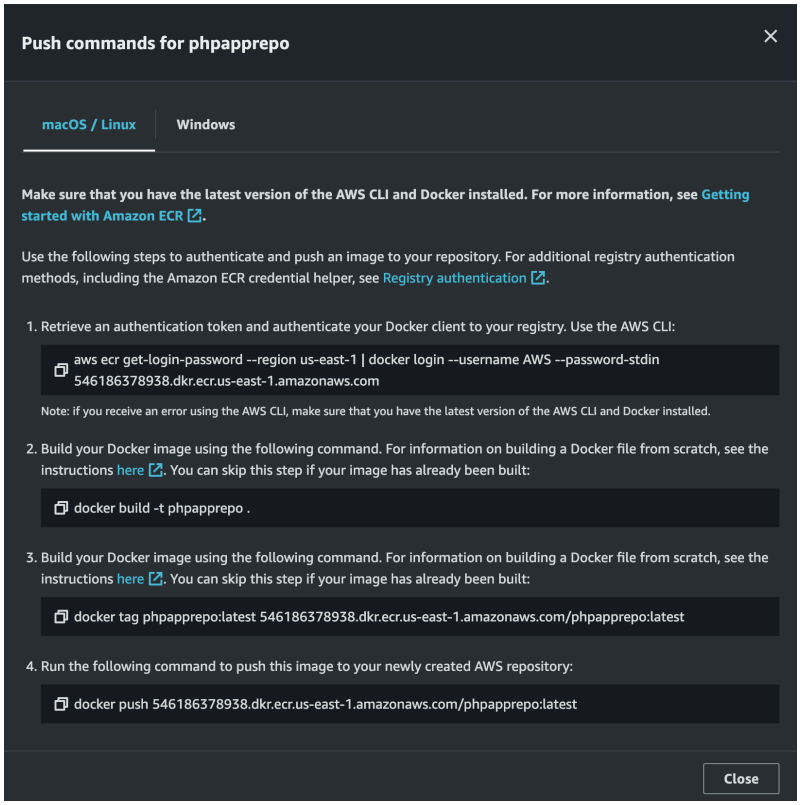
- AWS Secret Access Key [None]: fhsk
- Default region name [None]: us-east-1
- Default output format [None]: json
- Remove and add the LabProfile detail to ~/.aws/credentials using nano
- aws ec2 describe-instances
- Follow the ECR creation steps, then continue the below
- aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 546186378938.dkr.ecr.us-east-1.amazonaws.com
- Take care to be in application directory the place Dockerfile persists for example in this case /var/www/html
- docker build -t phpapprepo .
- docker tag phpapprepo:latest 546186378938.dkr.ecr.us-east-1.amazonaws.com/phpapprepo:latest
- docker push 546186378938.dkr.ecr.us-east-1.amazonaws.com/phpapprepo:latest

❖ ECR (**Amazon Elastic Container Registry**)

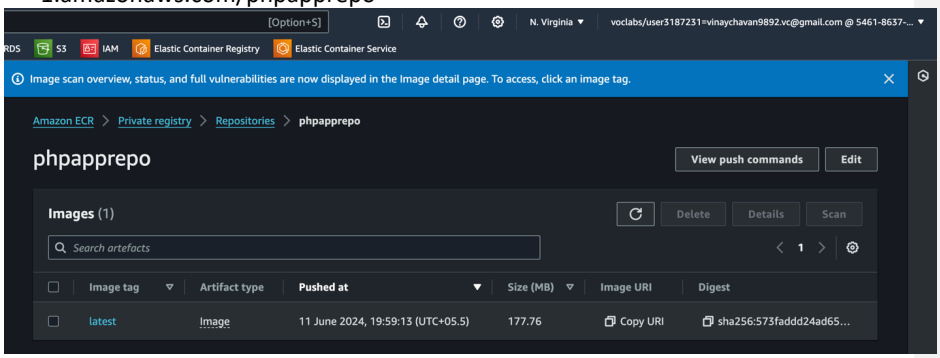
- Create repository
- Set visibility to private.
- Repository name:- 546186378938.dkr.ecr.us-east-1.amazonaws.com/phpapprepo
- rest of the setting as default.



- open repository and click on View command to follow further steps on Database container.

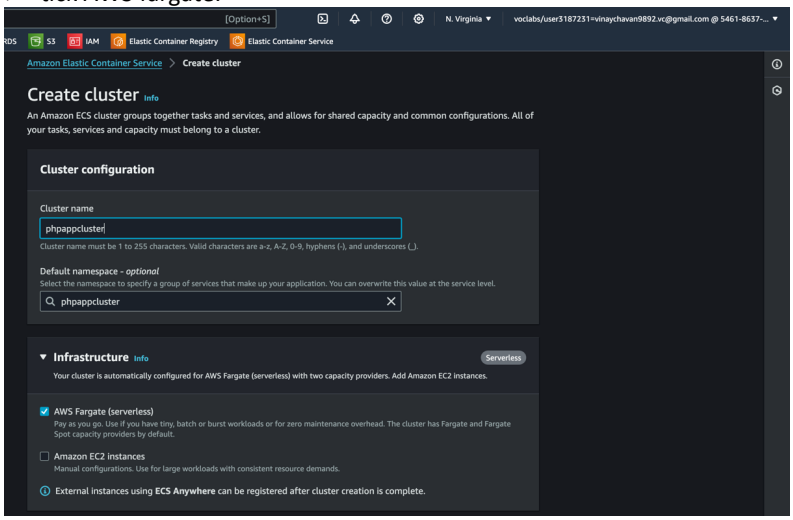


- after performing above steps.
- The image will be in repository.
- Kindly copy the repository uri:- 546186378938.dkr.ecr.us-east-1.amazonaws.com/phpapprepo



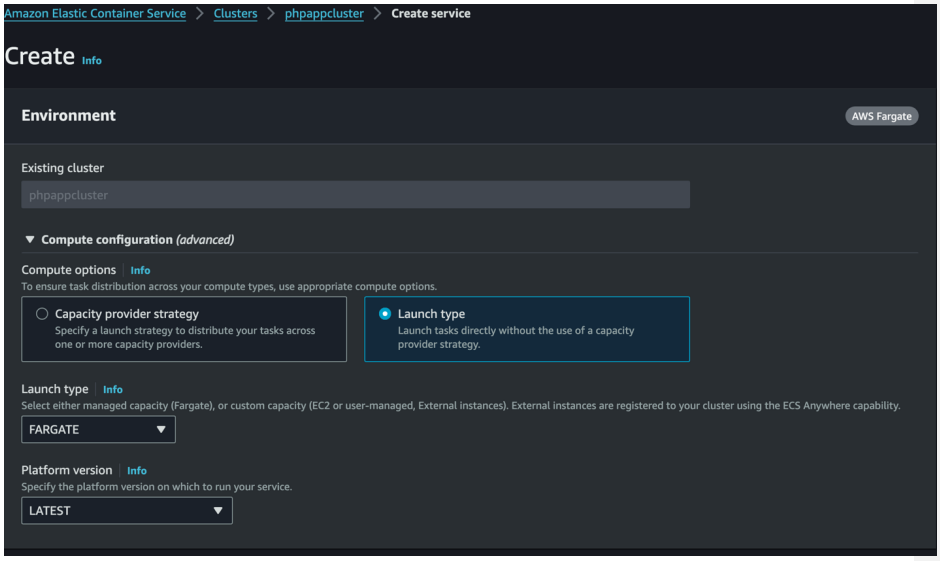
❖ ECS(**Amazon Elastic Container Service**)

- open ECS.
- Click on get started.
- Go to Cluster.
- Click on create cluster
- Set Cluster name.
- tick AWS fargate.

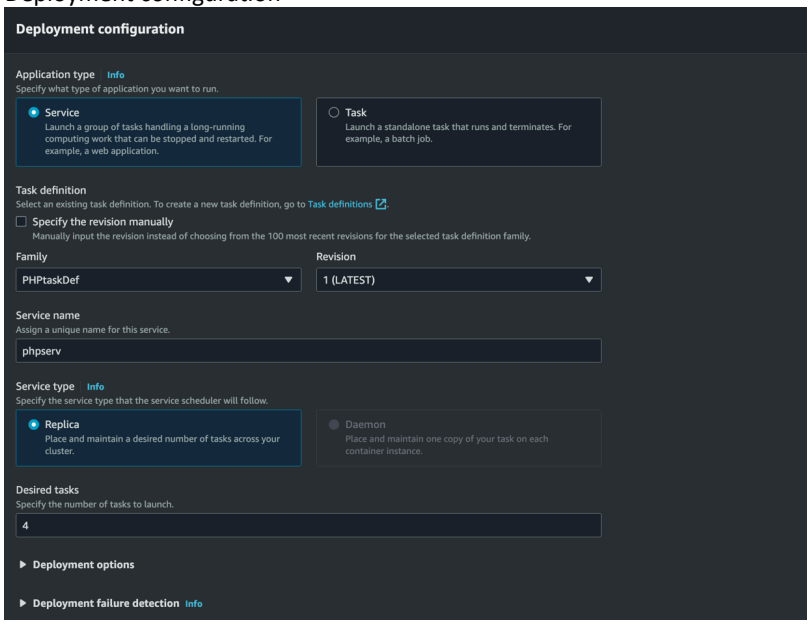


- Rest of the setting as default
- Click on create.
- The Cluster creation takes time.
- Now Click on “Task Definitions”.
- Click on “Create New task definition”.
- Set the name.
- Tick AWS fargate.
- Set Task Role and Task Execution Role to LabRole.
- Set the container name and past the URI of ECR(546186378938.dkr.ecr.us-east-1.amazonaws.com/phpapprepo).
- Add the below environmental variable.
- DB_HOST: (IP address of the EC2 instance running MySQL)
- DB_USER: root
- DB_PASSWORD: my-secret-pw
- DB_NAME: mydatabase
- Rest will be same as default.
- Now again go to the Cluster.
- Go to Services.
- Click on create service.

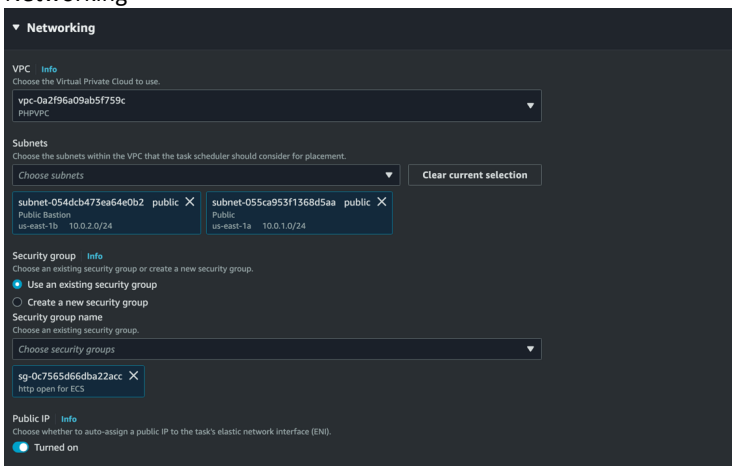
- Select launch type.
- Default as FARGATE and Latest.



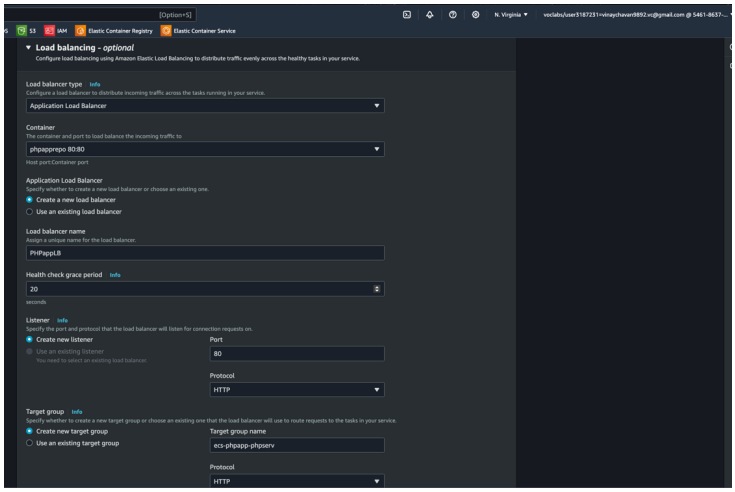
❖ Deployment configuration



❖ Networking

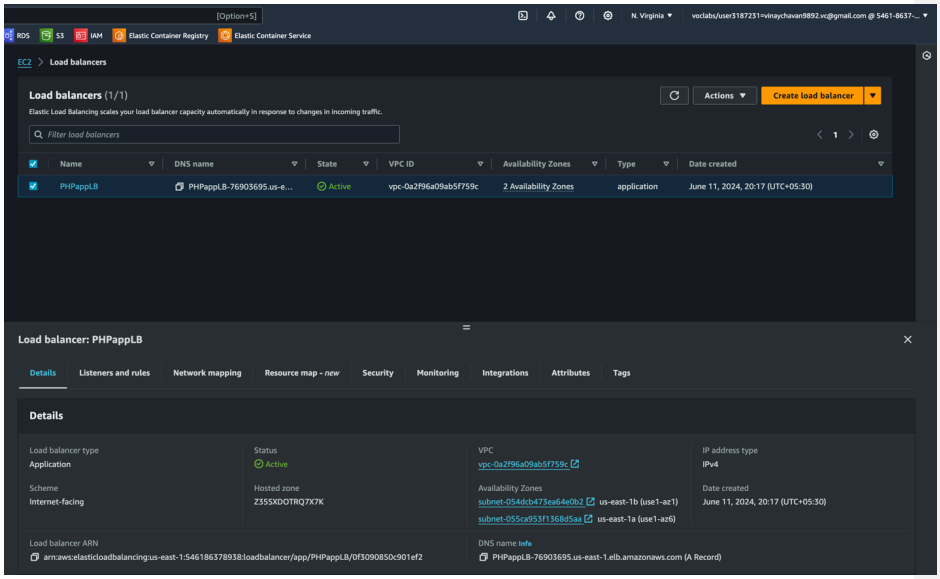


❖ Load Balancer

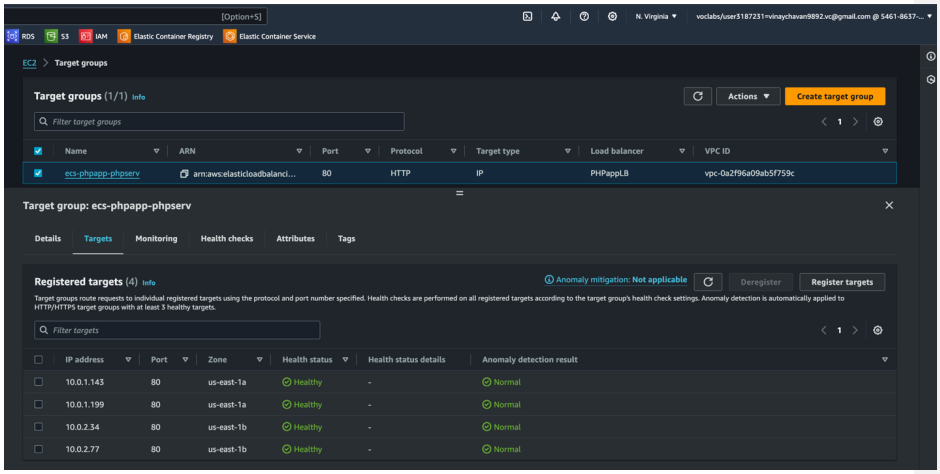


- Rest will be default.
- Click on create.
- Once created, use the LoadBalancer DNS name.

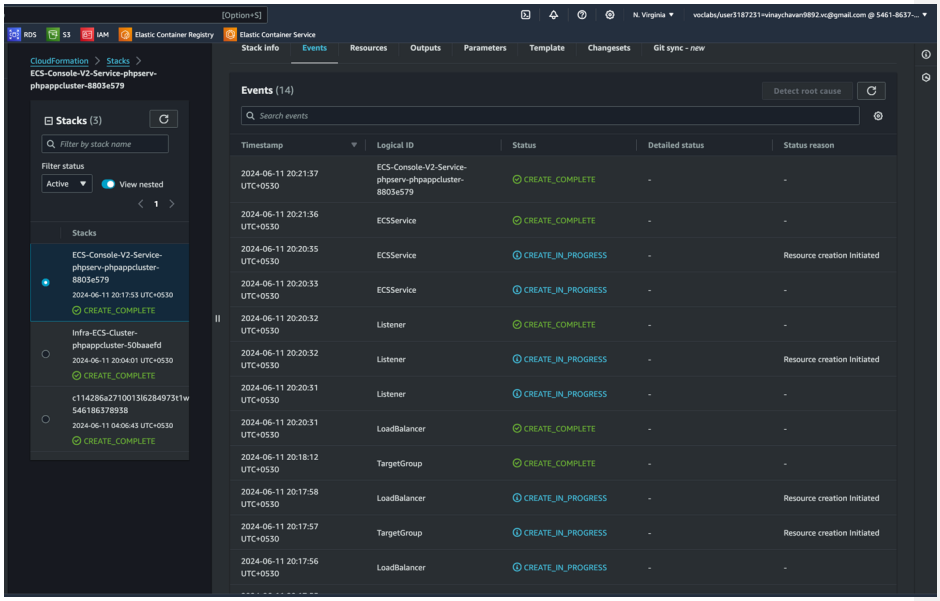
❖ Load Balancer



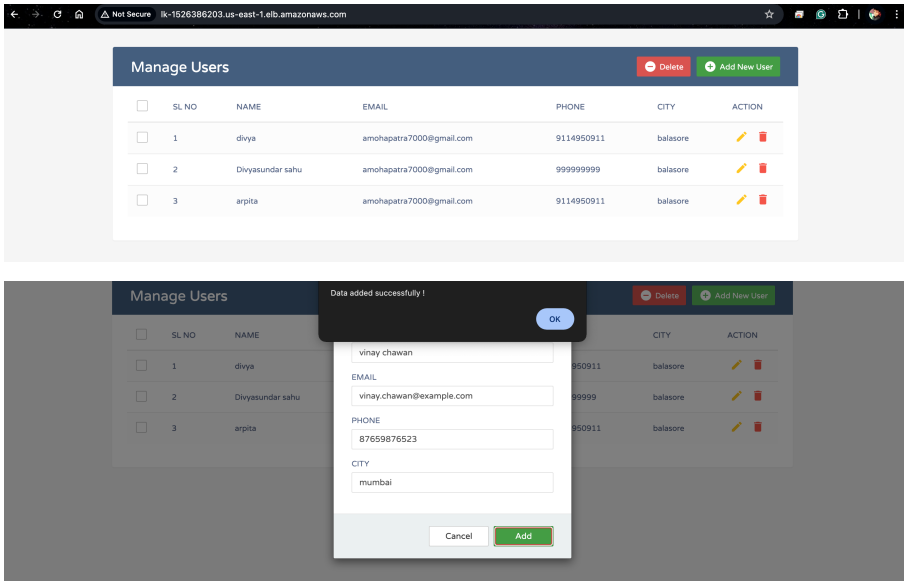
❖ Target Group

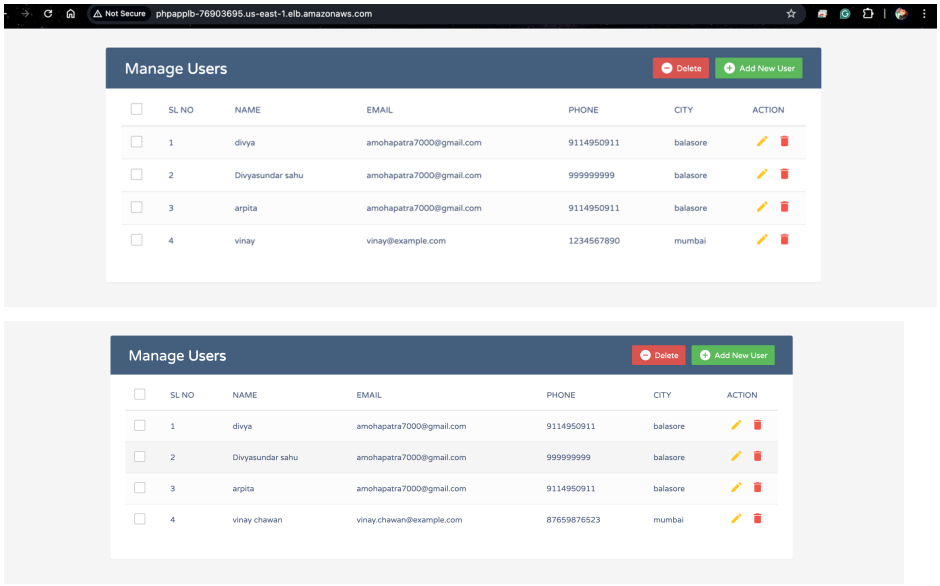


❖ CloudFormation(Status)



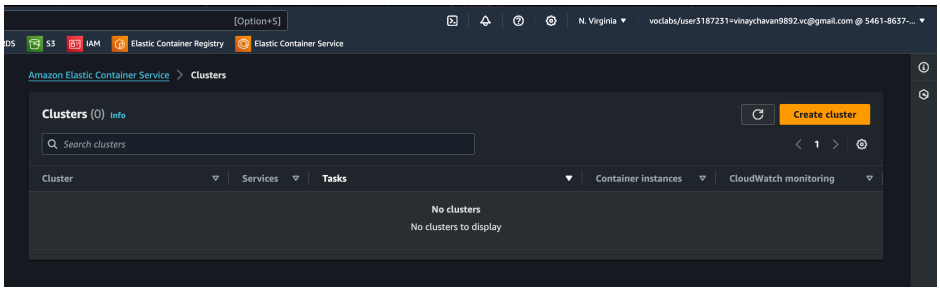
Final Output: - Did the addition of my name for testing.

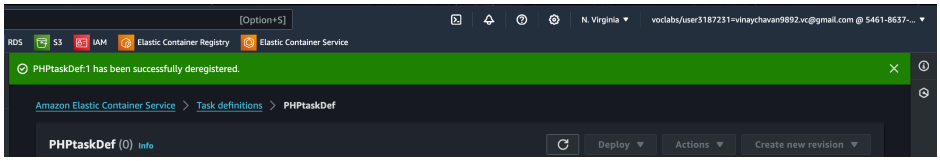


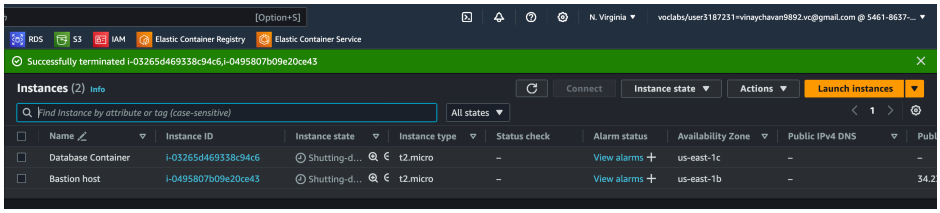

Not Secure
phpapplb-76903695.us-east-1.elb.amazonaws.com
Manage Users
Delete
Add New User
SL NO
NAME
EMAIL
PHONE
CITY
ACTION
1 divya amohapatra7000@gmail.com 9114950911 balasore
2 Divyasundar sahu amohapatra7000@gmail.com 999999999 balasore
3 arpita amohapatra7000@gmail.com 9114950911 balasore
4 vinay vinay@example.com 1234567890 mumbai

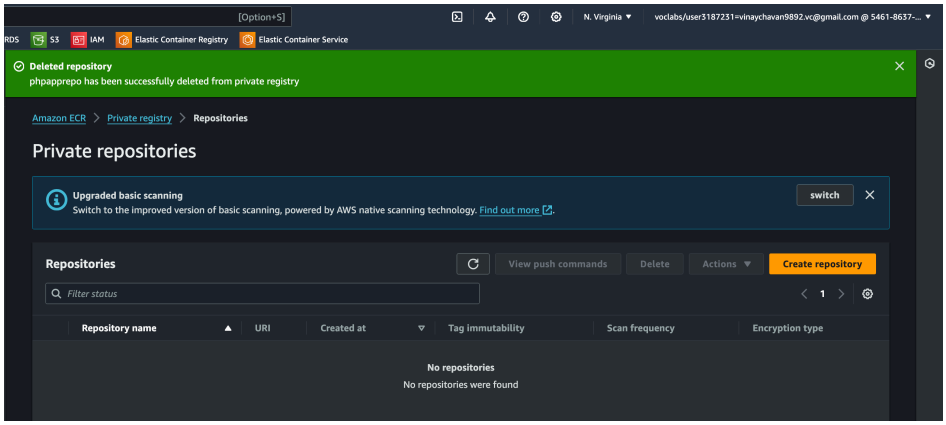
Manage Users
Delete
Add New User
SL NO
NAME
EMAIL
PHONE
CITY
ACTION
1 divya amohapatra7000@gmail.com 9114950911 balasore
2 Divyasundar sahu amohapatra7000@gmail.com 999999999 balasore
3 arpita amohapatra7000@gmail.com 9114950911 balasore
4 vinay chawan vinay.chawan@example.com 87659876523 mumbai

Deletion:-


[Option+S]
N. Virginia
S3
IAM
Elastic Container Registry
Elastic Container Service
Amazon Elastic Container Service
Clusters
Clusters (0) Info
Create cluster
Search clusters
Cluster
Services
Tasks
Container instances
CloudWatch monitoring
No clusters
No clusters to display


[Option+S]
N. Virginia
voclabs/user3187231=vinaychavan9892.vc@gmail.com @ 5461-8637-...
RDS
S3
IAM
Elastic Container Registry
Elastic Container Service
PHPtaskDef:1 has been successfully deregistered.
Amazon Elastic Container Service
Task definitions
PHPtaskDef
PHPtaskDef (0) Info
Deploy
Actions
Create new revision


[Option+S]
N. Virginia
voclabs/user3187231=vinaychavan9892.vc@gmail.com @ 5461-8637-...
RDS
S3
IAM
Elastic Container Registry
Elastic Container Service
Successfully terminated i-03265d469338c94c6,i-0495807b09e20ce43
Instances (2) Info
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4 DNS
Database Container
i-03265d469338c94c6
Shutting-d...
t2.micro
View alarms
us-east-1c
Bastion host
i-0495807b09e20ce43
Shutting-d...
t2.micro
View alarms
us-east-1b


[Option+S]
N. Virginia
voclabs/user3187231=vinaychavan9892.vc@gmail.com @ 5461-8637-...
RDS
S3
IAM
Elastic Container Registry
Elastic Container Service
Deleted repository
phpapprepo has been successfully deleted from private registry
Amazon ECR
Private registry
Repositories
Private repositories
Upgraded basic scanning
Switch to the improved version of basic scanning, powered by AWS native scanning technology. Find out more.
switch
Repositories
View push commands
Delete
Actions
Create repository
Filter status
Repository name
URI
Created at
Tag immutability
Scan frequency
Encryption type
No repositories
No repositories were found

THE END