

CONTAINER

Created By:- vinay chawan

Project Objectives and Requirements

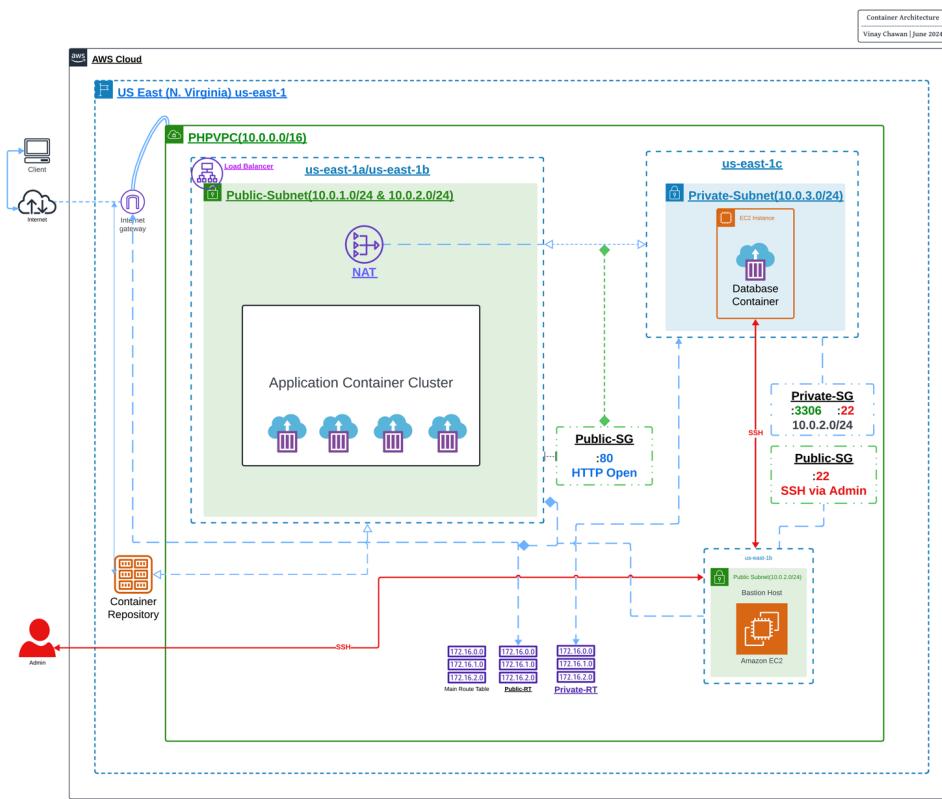
Prelude-From Infrastructure to Containers

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.

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:

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR	Available IPv
Public Bastion	subnet-054db47fea14e0b2	Available	vpc-0a2f96a09ab5f759c PHPVPC	10.0.2.0/24	-	251
Public	subnet-055ca953f1368d5aa	Available	vpc-0a2f96a09ab5f759c PHPVPC	10.0.1.0/24	-	251

Private Subnet:

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR	Available IPv
Private	subnet-0ba17a4e17f5baaf1	Available	vpc-0a2f96a09ab5f759c PHPVPC	10.0.3.0/24	-	251

❖ Route Table:

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

Public Route Table:

Name	Route table ID	Explicit subnet associations	Edge associations	Main	VPC	Owner ID
-	rtb-07416bd2a323f44a0	-	-	Yes	vpc-0ea379364rf64a9d	546186378938
-	rtb-03925caf53437547	-	-	Yes	vpc-0a2f96a09ab5f759c PHPV...	546186378938
public	rtb-04c1c248620311277	2 subnets	-	No	vpc-0a2f96a09ab5f759c PHPV...	546186378938
Private	rtb-0494b81013fa7ba9d	subnet-0ba17a4e17f5baaf1 / Private	-	No	vpc-0a2f96a09ab5f759c PHPV...	546186378938

rtb-04c1c248620311277 / public																		
Details	Routes	Subnet associations	Edge associations	Route propagation	Tags													
Explicit subnet associations (2) <table border="1"> <thead> <tr> <th>Name</th> <th>Subnet ID</th> <th>IPv4 CIDR</th> <th>IPv6 CIDR</th> </tr> </thead> <tbody> <tr> <td>Public Bastion</td> <td>subnet-054db47fea14e0b2</td> <td>10.0.2.0/24</td> <td>-</td> </tr> <tr> <td>Public</td> <td>subnet-055ca953f1368d5aa</td> <td>10.0.1.0/24</td> <td>-</td> </tr> </tbody> </table>							Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Public Bastion	subnet-054db47fea14e0b2	10.0.2.0/24	-	Public	subnet-055ca953f1368d5aa	10.0.1.0/24	-
Name	Subnet ID	IPv4 CIDR	IPv6 CIDR															
Public Bastion	subnet-054db47fea14e0b2	10.0.2.0/24	-															
Public	subnet-055ca953f1368d5aa	10.0.1.0/24	-															

Private Route Table:

The screenshot shows the AWS Route Tables page. At the top, there are navigation icons for RDS, S3, IAM, and Elastic Container Registry. The main header includes 'Route tables (1/4) Info' and 'Create route table'. Below the header is a search bar and a table with the following data:

Name	Route table ID	Explicit subnet associations	Main	VPC	Owner ID
-	rtb-07416bd2a323f44e0	-	Yes	vpc-0ee373564cf04a9d	546186378938
-	rtb-03975c1f5457e47	-	Yes	vpc-0a2f96a09ab5f759c PHPV...	546186378938
public	rtb-04c1c248630311277	2 subnets	No	vpc-0a2f96a09ab5f759c PHPV...	546186378938
Private	rtb-0494b81013fa7ba9d	subnet-0ba17a4e17f5baaf1 / Private	-	vpc-0a2f96a09ab5f759c PHPV...	546186378938

Below the table, a specific route table is selected: **rtb-0494b81013fa7ba9d / Private**. The details for this table are shown in a modal:

Route table ID	Main	Explicit subnet associations	Edge associations
rtb-0494b81013fa7ba9d	No	subnet-0ba17a4e17f5baaf1 / Private	-
VPC	Owner ID	vpc-0a2f96a09ab5f759c PHPVPC 546186378938	

❖ Internet Gateways (IGW)

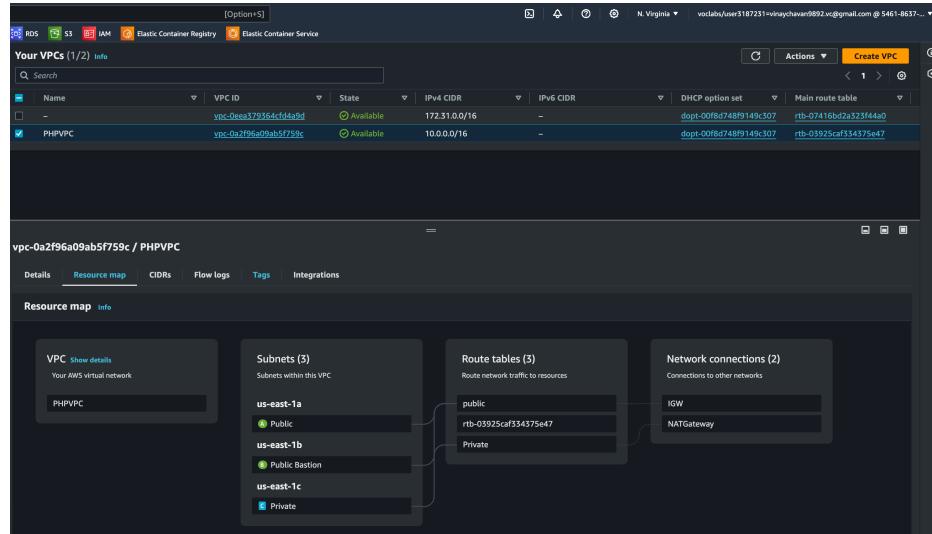
- Click on Internet Gateways.
- Click on “Internet gateway”.
- Name and tag the internet gateway as per requirement.
- Click on Create Internet gateway.
- Click on “Attach to a VPC”
- Select the OwnCloud-VPC.
- Click on Attach internet gateway.
- Click on Route tables.
- Select “Public” route table.
- 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.

The screenshot shows the AWS CloudWatch Metrics console. At the top, there are navigation links for RDS, SNS, IAM, and Elastic Container Registry. The main area displays a message: "Security groups for eni-09a3ae5774a16750 changed successfully". Below this, a table titled "Instances (2) Info" lists two instances: "Database Container" and "Bastion host". Both instances are running, t2.micro type, in us-east-1c, with 2/2 checks passed. A "Launch Instances" button is visible at the top right.

The screenshot shows the AWS EC2 Instances page. It lists two instances: "Database Container" and "Bastion host". The "Bastion host" instance is selected. Below the table, a detailed view for the "Bastion host" instance (i-0495807b09e20ce43) is shown. The "Details" tab is selected, displaying various configuration parameters:

- Instance ID: i-0495807b09e20ce43 (Bastion host)
- Public IPv4 address: 34.234.211.89 (with a "View address" link)
- Private IPv4 address: 10.0.2.16
- Instance state: Running
- Private IP DNS name: ip-10-0-2-16.ec2.internal
- Instance type: t2.micro
- VPC ID: vpc-0a2f96a09ab5f759c (PHPVPC)
- Subnet ID: subnet-0544d477 (with a "View details" link)
- Auto Scaling Group name: auto-scaling-group-12345678901234567890

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:

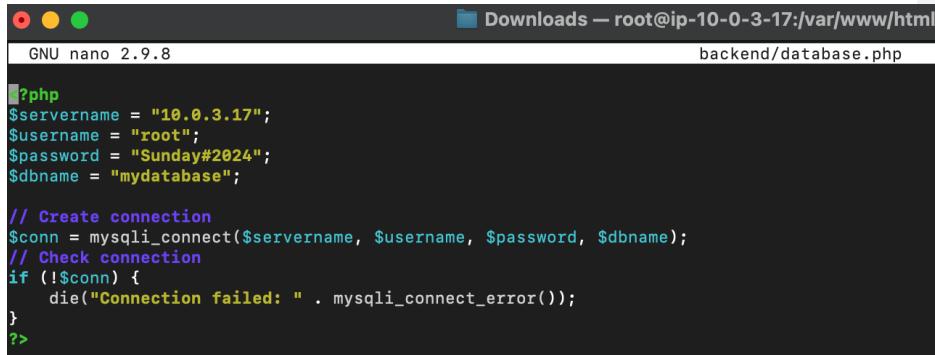
- Since the Database is in a Private Network by using Bastion host ,will need to take SSH.
- Hence need to send .pem to Database.
- Run this command, to send the Database.pem to Database Container.

```
scp -i bastion.pem ./database.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
```

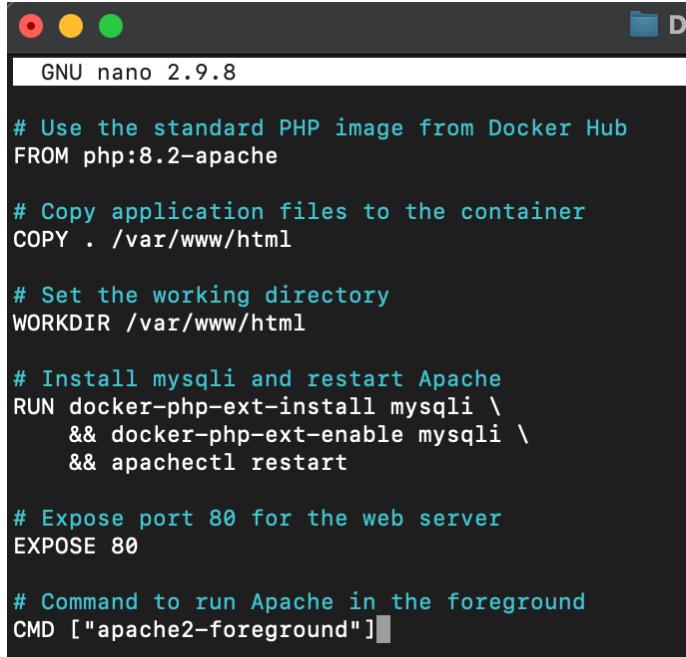


```
GNU nano 2.9.8 backend/database.php

?php
$servername = "10.0.3.17";
$username = "root";
$password = "Sunday#2024";
$dbname = "mydatabase";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
?>
```

```
➤ nano Dockerfile
```



```
GNU nano 2.9.8 Dockerfile

# Use the standard PHP image from Docker Hub
FROM php:8.2-apache

# Copy application files to the container
COPY . /var/www/html

# Set the working directory
WORKDIR /var/www/html

# Install mysqli and restart Apache
RUN docker-php-ext-install mysqli \
    && docker-php-ext-enable mysqli \
    && apachectl restart

# Expose port 80 for the web server
EXPOSE 80

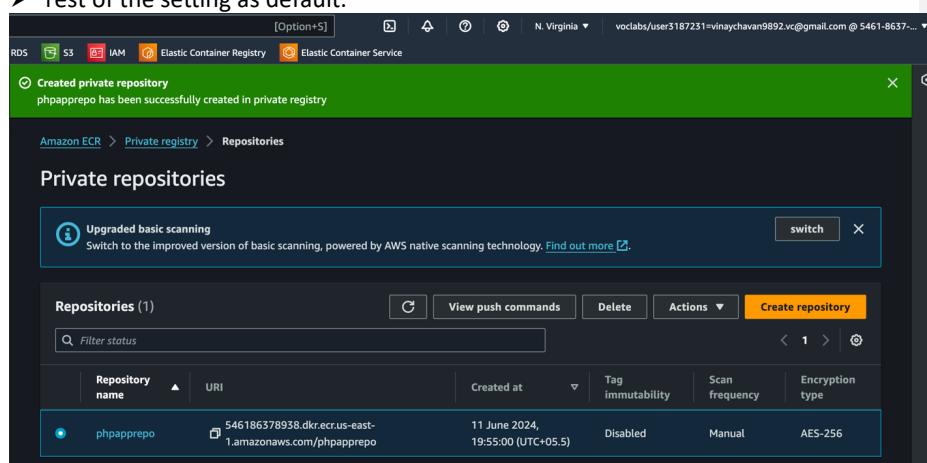
# Command to run Apache in the foreground
CMD ["apache2-foreground"]
```

```
➤ 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

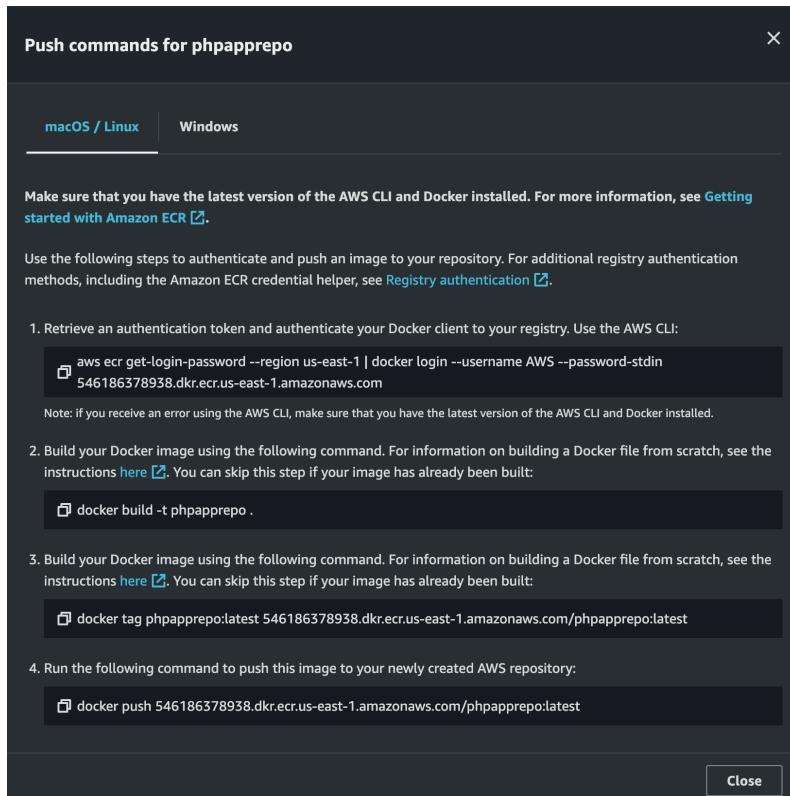
❖ **ECR (Amazon Elastic Container Registry)**

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



The screenshot shows the AWS ECR console interface. At the top, there are navigation links for RDS, S3, IAM, and Elastic Container Registry (which is highlighted). On the right, it shows the user's email and session information. A prominent green notification bar at the top center says "Created private repository" and "phpapprepo has been successfully created in private registry". Below this, the breadcrumb navigation shows "Amazon ECR > Private registry > Repositories". The main title is "Private repositories". A message box indicates "Upgraded basic scanning" with a link to "Find out more". The main table lists one repository: "phpapprepo" with URI "546186378938.dkr.ecr.us-east-1.amazonaws.com/phpapprepo". The table includes columns for Repository name, URI, Created at, Tag immutability, Scan frequency, and Encryption type. The "Scan frequency" is listed as "Manual" and "Encryption type" as "AES-256".

- 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

Amazon ECR > Private registry > Repositories > **phpapprepo**

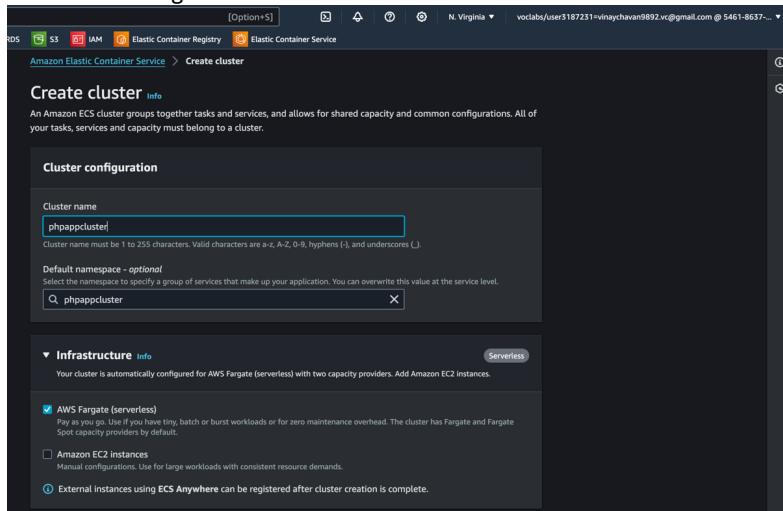
phpapprepo

Images (1)

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
latest	Image	11 June 2024, 19:59:13 (UTC+05.5)	177.76	Copy URI	sha256:573fadd24ad65...

❖ 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.

The screenshot shows the 'Create' page for a service in the AWS Elastic Container Service. The 'Compute configuration (advanced)' section is expanded. Under 'Launch type', the 'Launch type' radio button is selected, which is highlighted with a blue border. The 'Capacity provider strategy' radio button is also visible but not selected. Below the launch type section, there are dropdown menus for 'Platform version' set to 'LATEST' and 'Launch type' set to 'FARGATE'. The top right corner of the interface has a 'AWS Fargate' button.

❖ Deployment configuration

The screenshot shows the 'Deployment configuration' page. Under 'Application type', the 'Service' radio button is selected. In the 'Task definition' section, the 'Specify the revision manually' checkbox is checked, and the 'Revision' dropdown is set to '1 (LATEST)'. The 'Service name' field contains 'phpserv'. Under 'Service type', the 'Replica' radio button is selected. In the 'Desired tasks' section, the number '4' is entered. At the bottom, there are sections for 'Deployment options' and 'Deployment failure detection'.

❖ Networking

The screenshot shows the 'Networking' configuration section for a Lambda function. It includes fields for VPC (selected VPC: vpc-0a2f96a09ab5f759c), Subnets (selected subnets: subnet-054dc473ea64e0b2 and subnet-055ca953f1568d5aa), Security group (selected security group: sg-0c7565d66dba22acc), and Public IP (set to 'Turned on').

❖ Load Balancer

The screenshot shows the 'Load balancing - optional' configuration for a Lambda function. It includes fields for Container (Container: phpapp:php:8080), Application Load Balancer (selected load balancer: alb-123456789012345678), Load balancer name (name: PvPqgtB), Health check grace period (value: 20 seconds), Listener (Port: 80, Protocol: HTTP), and Target group (selected target group: ecs-phiapp-phiapp).

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

❖ Load Balancer

The screenshot shows the AWS Load Balancers console. At the top, there are tabs for RDS, S3, IAM, Basic Container Registry, and Elastic Container Service. The main heading is "Load balancers (1/1)". Below this is a table with one row, showing the details of the "PHPappLB" load balancer. The table columns include Name, DNS name, State, VPC ID, Availability Zones, Type, and Date created. The "PHPappLB" row has "Active" in the State column, "vpc-0a2f96a09ab5f759c" in the VPC ID column, "2 Availability Zones" in the Availability Zones column, "application" in the Type column, and "June 11, 2024, 20:17 (UTC+05:30)" in the Date created column. Below the table, a modal window titled "Load balancer: PHPappLB" is open, showing the "Details" tab. It displays the load balancer type as "Application", the VPC as "vpc-0a2f96a09ab5f759c", and the IP address type as "IPv4". The scheme is listed as "Internet-facing". The hosted zone is "Z555XOCYRQ7X7K". The ARN is "arn:aws:elasticloadbalancing:us-east-1:546186378958:loadbalancer/app/PHPappLB/0f3090850c901ef2". The DNS name is "PHPappLB-76903695.us-east-1.elb.amazonaws.com (A Record)".

❖ Target Group

The screenshot shows the AWS Target groups console. At the top, there are tabs for RDS, S3, IAM, Basic Container Registry, and Elastic Container Service. The main heading is "Target groups (1/1)". Below this is a table with one row, showing the details of the "ecs-phapp-phpserv" target group. The table columns include Name, ARN, Port, Protocol, Target type, Load balancer, and VPC ID. The "ecs-phapp-phpserv" row has "80" in the Port column, "HTTP" in the Protocol column, "IP" in the Target type column, "PHPappLB" in the Load balancer column, and "vpc-0a2f96a09ab5f759c" in the VPC ID column. Below the table, a modal window titled "Target group: ecs-phapp-phpserv" is open, showing the "Targets" tab. It displays four registered targets with their IP addresses, ports, zones, health status, and anomaly detection results. All targets are marked as healthy and normal.

IP address	Port	Zone	Health status	Anomaly detection result
10.0.1.145	80	us-east-1a	Healthy	Normal
10.0.1.199	80	us-east-1a	Healthy	Normal
10.0.2.34	80	us-east-1b	Healthy	Normal
10.0.2.77	80	us-east-1b	Healthy	Normal

❖ CloudFormation(Status)

The screenshot shows the AWS CloudFormation Events page. The left sidebar displays the stack structure, including the main stack 'ECS-Console-V2-Service-phpserv-phpappcluster-8805e579' and its nested stacks 'Infra-ECS-Cluster-phpprovider-50baef' and 'c1428627100136284973t1w'. The main content area shows a table of events:

Timestamp	Logical ID	Status	Detailed status	Status reason
2024-06-11 20:21:37 UTC+0530	ECS-Console-V2-Service-phpserv-phpappcluster-8805e579	CREATE_COMPLETE	-	-
2024-06-11 20:21:36 UTC+0530	ECSERVICE	CREATE_COMPLETE	-	-
2024-06-11 20:20:35 UTC+0530	ECSERVICE	CREATE_IN_PROGRESS	-	Resource creation initiated
2024-06-11 20:20:33 UTC+0530	ECSERVICE	CREATE_IN_PROGRESS	-	-
2024-06-11 20:20:32 UTC+0530	Listener	CREATE_COMPLETE	-	-
2024-06-11 20:20:32 UTC+0530	Listener	CREATE_IN_PROGRESS	-	Resource creation initiated
2024-06-11 20:20:31 UTC+0530	Listener	CREATE_IN_PROGRESS	-	-
2024-06-11 20:20:31 UTC+0530	LoadBalancer	CREATE_COMPLETE	-	-
2024-06-11 20:18:12 UTC+0530	TargetGroup	CREATE_COMPLETE	-	-
2024-06-11 20:17:58 UTC+0530	LoadBalancer	CREATE_IN_PROGRESS	-	Resource creation initiated
2024-06-11 20:17:57 UTC+0530	TargetGroup	CREATE_IN_PROGRESS	-	Resource creation initiated
2024-06-11 20:17:56 UTC+0530	LoadBalancer	CREATE_IN_PROGRESS	-	-

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

The screenshots show the 'Manage Users' interface. The top screenshot displays a list of existing users:

SL.NO	NAME	EMAIL	PHONE	CITY	ACTION
1	divya	amohipatral7000@gmail.com	9114950911	balasore	
2	Divyashundar sahu	amohipatral7000@gmail.com	9999999999	balasore	
3	arpita	amohipatral7000@gmail.com	9114950911	balasore	

The bottom screenshot shows a modal dialog for adding a new user, with the message 'Data added successfully!' displayed above the input fields:

Manage Users

Data added successfully!

SL NO	NAME	EMAIL	PHONE	CITY	ACTION
1	divya	amohipatral7000@gmail.com	9114950911	balasore	
2	Divyashundar sahu	amohipatral7000@gmail.com	9999999999	balasore	
3	arpita	amohipatral7000@gmail.com	9114950911	balasore	

vinay chawan
EMAIL: vinay.chawan@example.com
PHONE: 87659876523
CITY: mumbai

OK Cancel Add

Manage Users						
	SL NO	NAME	EMAIL	PHONE	CITY	ACTION
<input type="checkbox"/>	1	divya	amohipatra7000@gmail.com	9114950911	balasore	
<input type="checkbox"/>	2	Divyasundar sahu	amohipatra7000@gmail.com	9999999999	balasore	
<input type="checkbox"/>	3	arpita	amohipatra7000@gmail.com	9114950911	balasore	
<input type="checkbox"/>	4	vinay	vinay@example.com	1234567890	mumbai	

Manage Users						
	SL NO	NAME	EMAIL	PHONE	CITY	ACTION
<input type="checkbox"/>	1	divya	amohipatra7000@gmail.com	9114950911	balasore	
<input type="checkbox"/>	2	Divyasundar sahu	amohipatra7000@gmail.com	9999999999	balasore	
<input type="checkbox"/>	3	arpita	amohipatra7000@gmail.com	9114950911	balasore	
<input type="checkbox"/>	4	vinay chawan	vinay.chawan@example.com	87659876523	mumbai	

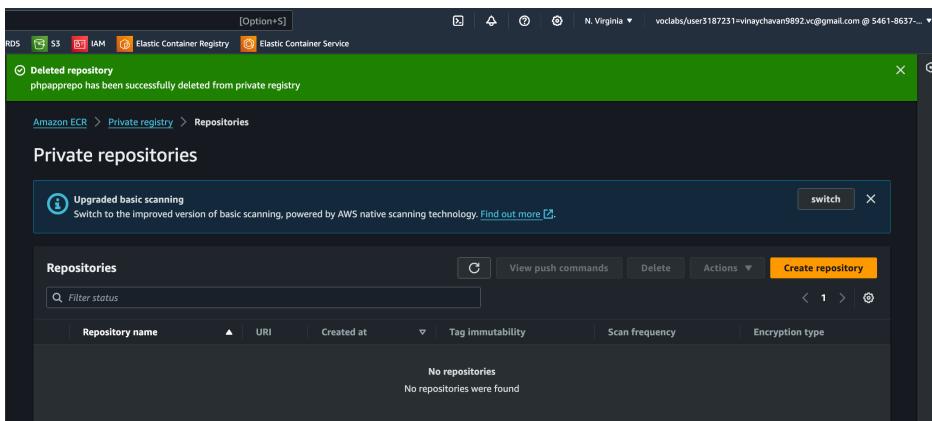
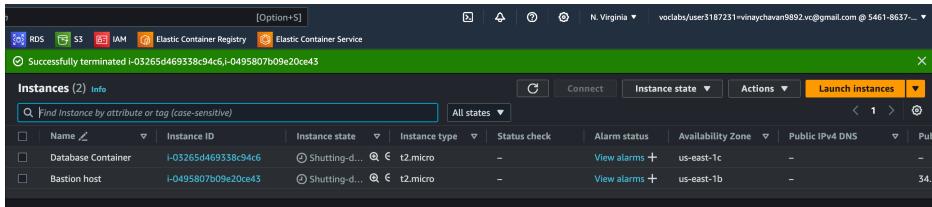
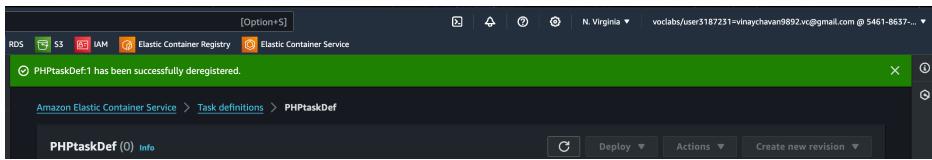
Deletion:-

Amazon Elastic Container Service > Clusters

Clusters (0) info

No clusters
No clusters to display

Create cluster



THE END