

# FULLSTACK APPLICATION DEPLOYMENT ON ECS-FARGATE THROUGH TERRAFORM

**1. Overview of Three-Tier Architecture** : Deploying an ECS task on private subnets ensures that your containers remain isolated from direct internet access while still being able to communicate internally within your VPC. Here's a step-by-step guide to deploying an ECS task in private subnets.

A three-tier architecture consists of:

**1. Presentation Layer (Frontend)**

- This is the user interface (UI) of the application, such as a web or mobile frontend.
- Deployed on **ECS (Fargate or EC2) running containers** with Nginx, Apache, or a React/Angular application.

**2. Application Layer (Backend)**

- This layer handles business logic and processing.
- Deployed on **ECS (Fargate or EC2) running containers** with Node.js, Java, Python, or another backend service.

**3. Data Layer (Database)**

- A managed relational database, such as **Amazon RDS (MySQL, PostgreSQL, or Aurora)**.
- Stores and manages application data.

**Key AWS Services Used**

- **Amazon ECS (Fargate or EC2)**: Manages containerized workloads.
  - **Elastic Load Balancer (ALB)**: Routes traffic to frontend/backend services.
  - **AWS RDS**: Managed relational database service.
  - **Amazon VPC**: Segregates frontend, backend, and database tiers into subnets.
  - **IAM & Security Groups**: Controls access between different layers.
- 
- 

- Create a ec2

Instance type

t2.micro

Free tier eligible

Family: t2

1 vCPU

1 GB Memory

Current generations: t2a1

On-Demand Linux base pricing: 0.0124 USD per hour

On-Demand Windows base pricing: 0.017 USD per hour

On-Demand RHEL base pricing: 0.0264 USD per hour

On-Demand Ubuntu base pricing: 0.0142 USD per hour

On-Demand SUSE base pricing: 0.0124 USD per hour

Additional costs apply for AMIs with pre-installed software

All generations

Configure instance types

▼ Key pair (login)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

mykeypair

Create new key pair

▼ Network settings

Network

vpc-086f42d7a241f9401

Subnet

No preference (Default subnet in any availability zone)

▼ Summary

Number of instances

1

Software image (AMI)

Amazon Linux 2023 AMI 2023.6.2...read more

Virtual server type (instance type)

t2.micro

Firewall (security group)

default

Storage (volumes)

1 volume(s) - 8 GB

Free tier: In your first year includes 750 hours of t2.micro or t3.micro in the Regions in which t2.micro is unavailable; instance usage on free tier after your month 120 hours of on-demand t2.micro

Cancel

Launch instance

Preview code

## ■ Attach iam role to ec2

Instances (1/1)

Last updated less than a minute ago

Connect

Instance state

Actions

Launch instance

Find instance by attribute or tag (case-sensitive)

All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm
ecs	i-077a185388b096543	Running	t2.micro	Initializing	View all

Change security groups

Get Windows password

Modify IAM role

Connect

View details

Manage instance state

Instance settings

Networking

Security

Image and templates

Monitor and troubleshoot

## ■ Update the role

EC2

Instances

i-077a185388b096543

Modify IAM role

Modify IAM role

Attach an IAM role to your instance.

Instance ID

i-077a185388b096543 (ecs)

IAM role

Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

ec2-admin

Create new IAM role

Cancel

Update IAM role

## ■ Connect to the ec2



- Install the git and docker by following commands

```
sudo yum install git -y
```

```
sudo yum install docker -y #linux 2023
```

```
sudo usermod -aG docker ec2-user
```

```
newgrp docker
```

```
sudo service docker start
```

```
sudo chmod 777 /var/run/docker.sock
```

```
[ec2-user@ip-172-31-7-58 ~]$ sudo yum install git -y
sudo yum install docker -y #linux 2023
sudo usermod -aG docker ec2-user
newgrp docker
sudo service docker start
sudo chmod 777 /var/run/docker.sock
```

- Clone the git repo for docker image build purpose

```
git clone https://github.com/CloudTechDevOps/2nd10WeeksofCloudOps-main.git
```

```
[ec2-user@ip-172-31-7-58 ~]$ git clone https://github.com/CloudTechDevOps/2nd10WeeksofCloudOps-main.git
Cloning into '2nd10WeeksofCloudOps-main'...
remote: Enumerating objects: 405, done.
remote: Counting objects: 100% (250/250), done.
remote: Compressing objects: 100% (109/109), done.
remote: Total 405 (delta 213), reused 141 (delta 141), pack-reused 155 (from 1)
Receiving objects: 100% (405/405), 3.66 MiB | 4.76 MiB/s, done.
Resolving deltas: 100% (239/239), done.
[ec2-user@ip-172-31-7-58 ~]$ ls
2nd10WeeksofCloudOps-main
[ec2-user@ip-172-31-7-58 ~]$
```

- Switch to client directory

cd 2nd10WeeksofCloudOps-main/client

```
[ec2-user@ip-172-31-7-58 ~]$ cd 2nd10WeeksofCloudOps-main/client/
[ec2-user@ip-172-31-7-58 client]$ ls
Dockerfile package-lock.json package.json public src
[ec2-user@ip-172-31-7-58 client]$
```

- Change the api base url according to the your requirement

vi src/pages/config.js

```
[ec2-user@ip-172-31-7-58 client]$ cat src/pages/config.js
// const API_BASE_URL = "http://localhost:8800";
const API_BASE_URL = "http://veera.narni.co.in";
export default API_BASE_URL;
[ec2-user@ip-172-31-7-58 client]$
```

- Create a ecr repository named as frontend

#### Create private repository

General settings

Repository name

Provide a concise name. Repository names support namespaces, which is recommended for grouping similar repositories.

545009827818.dkr.ecr.ap-south-1.amazonaws.com/frontend

It can't be 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, and special characters \_ and -.

Image tag mutability

Specify the tag mutability setting to use. When tag immutability is turned on for a repository, tags are prevented from being overwritten.

☒ Mutable  
Image tags can be overwritten.

☐ Immutable  
Image tags are prevented from being overwritten.

Encryption

The encryption settings for a repository can't be changed once the repository is created.

Encryption configuration

- Open the frontend ecr and click on **view push commands**



- Copy all commands paste it on ec2 one by one in client directory
- Copy the 1<sup>st</sup> login command and paste it on ec2 client directory

[macOS / Linux](#)
[Windows](#)

Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see [Getting Started with Amazon ECR](#).

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see [Registry Authentication](#).

- Retrieve an authentication token and authenticate your Docker client to your registry. Use the AWS CLI:
 

```
aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 545009827818.dkr.ecr.ap-south-1.amazonaws.com
```

Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.
- Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions [here](#). You can skip this step if your image is already built:
 

```
docker build -t frontend .
```
- After the build completes, tag your image so you can push the image to this repository:
 

```
docker tag frontend:latest 545009827818.dkr.ecr.ap-south-1.amazonaws.com/frontend:latest
```
- Run the following command to push this image to your newly created AWS repository:
 

```
docker push 545009827818.dkr.ecr.ap-south-1.amazonaws.com/frontend:latest
```

Close

- Copy the 1<sup>st</sup> login command and paste it on ec2 client directory

```
[ec2-user@ip-172-31-7-58 client]$ aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 545009827818.dkr.ecr.ap-south-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[ec2-user@ip-172-31-7-58 client]$
```

- Copy the 2<sup>nd</sup> build command and paste it on ec2 client directory

```
[ec2-user@ip-172-31-7-58 client]$ docker build -t frontend .
[+] Building 0.9s (1/3)
```

- Copy the 3<sup>rd</sup> tag command and paste it on ec2 client directory

```
[ec2-user@ip-172-31-7-58 client]$ docker tag frontend:latest 545009827818.dkr.ecr.ap-south-1.amazonaws.com/frontend:latest
[ec2-user@ip-172-31-7-58 client]$
```

- Copy the 4<sup>th</sup> push command and paste it on ec2 client directory

```
[ec2-user@ip-172-31-7-58 client]$ docker push 545009827818.dkr.ecr.ap-south-1.amazonaws.com/frontend:latest
The push refers to repository [545009827818.dkr.ecr.ap-south-1.amazonaws.com/frontend]
afa9a0ec131e: Pushed
e48683950315: Pushed
1b533b3f600d: Pushed
45d2a6f2a0b1: Pushed
5f70bf18a086: Pushed
d465f9c6793b: Pushed
7914c8f600f5: Pushed
latest: digest: sha256:72207fd0031deee427ffa317efb07d74dca1473a8efb82eb7d09f3733d990 size: 1782
[ec2-user@ip-172-31-7-58 client]$
```

- Frontend image is pushed into ecr successfully



- Switch to backend directory

```
[ec2-user@ip-172-31-7-58 client]$ cd ..
[ec2-user@ip-172-31-7-58 2nd10WeeksofCloudOps-main]$ cd backend/
[ec2-user@ip-172-31-7-58 backend]$ ls
Dockerfile index.js package-lock.json package.json test.sql
[ec2-user@ip-172-31-7-58 backend]$
```

- Create ecr repo for backend named as backend



- Copy all commands paste it on ec2 one by one in backend directory
- Copy the 1<sup>st</sup> login command and paste it on ec2 backend directory

Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see [Getting Started with Amazon ECR](#).

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see [Registry Authentication](#).

1. Retrieve an authentication token and authenticate your Docker client to your registry. Use the AWS CLI:

```
aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 545009827818.dkr.ecr.ap-south-1.amazonaws.com
```

Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.

2. Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions [here](#). You can skip this step if your image is already built:

```
docker build -t backend .
```

3. After the build completes, tag your image so you can push the image to this repository:

```
docker tag backend:latest 545009827818.dkr.ecr.ap-south-1.amazonaws.com/backend:latest
```

4. Run the following command to push this image to your newly created AWS repository:

```
docker push 545009827818.dkr.ecr.ap-south-1.amazonaws.com/backend:latest
```

- Copy the 2<sup>nd</sup> build command and paste it on ec2 backend directory

```
[ec2-user@ip-172-31-7-58 backend]$ docker build -t backend .  
[+] Building 0.3s (1/2)  
=> [internal] load build definition from Dockerfile  
=> => transferring dockerfile: 738B  
=> [internal] load metadata for docker.io/library/node:18
```

- Copy the 3<sup>rd</sup> tag command and paste it on ec2 backend directory

```
[ec2-user@ip-172-31-7-58 backend]$ docker tag backend:latest 545009827818.dkr.ecr.ap-south-1.amazonaws.com/backend:latest  
[ec2-user@ip-172-31-7-58 backend]$
```

- Copy the 4<sup>th</sup> push command and paste it on ec2 backend directory

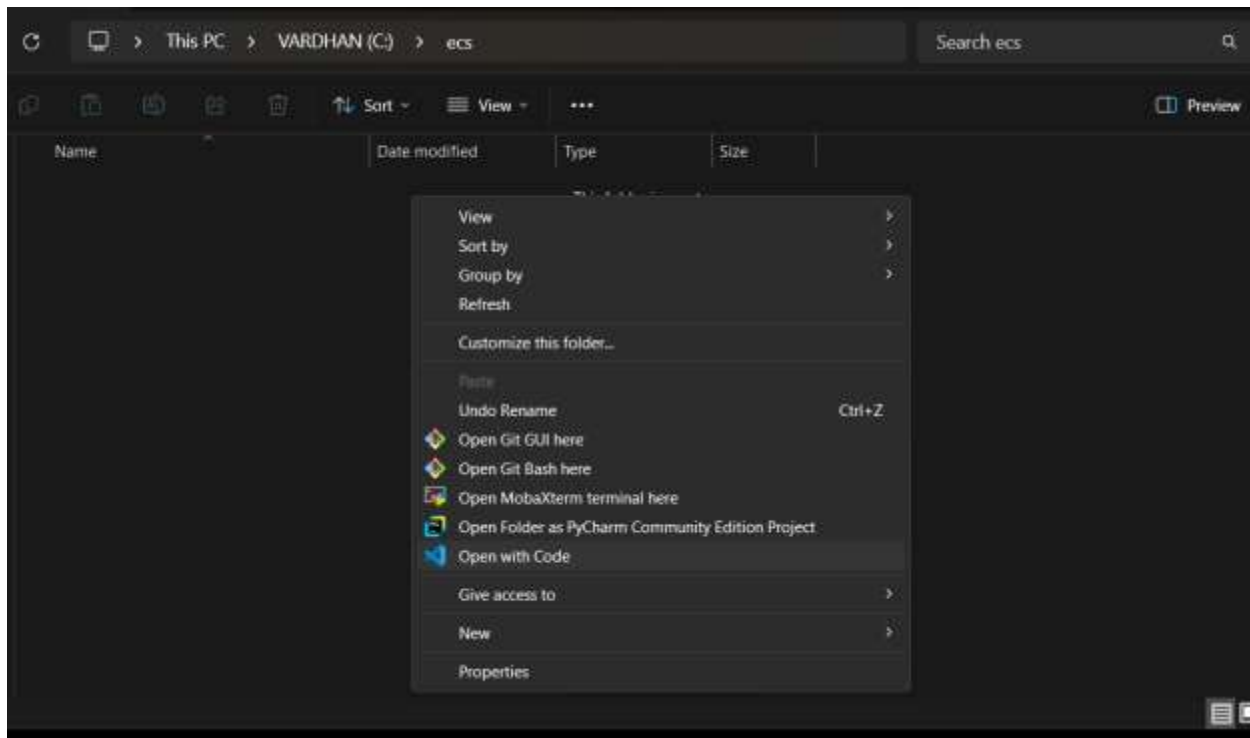
```
[ec2-user@ip-172-31-7-58 backend]$ docker push 545009827818.dkr.ecr.ap-south-1.amazonaws.com/backend:latest  
The push refers to repository 1545009827818.dkr.ecr.ap-south-1.amazonaws.com/backend  
7626fab9e78f: Pushed  
b65677ed5817: Pushed  
4b1963b1f5c4: Pushed  
b54ab579bea8: Pushed  
8d3309036362: Pushed  
bcfca6d69e55: Pushed  
f1f62a16538c: Pushed  
48b86cff2099: Pushed  
48cc25dc4131: Pushed  
da1b0a0d9f7f: Pushed  
41d4dc7516bb: Pushed  
c0f51b8dc37d: Pushed  
91b542912d12: Pushed  
latest: digest: sha256:64bf82e92b7403d2c66c1d3cfa2aeadd7523f2f47124e103db4817c9ebb9c591 size: 3052  
[ec2-user@ip-172-31-7-58 backend]$
```



- Pushed the backend image to ecr successfully



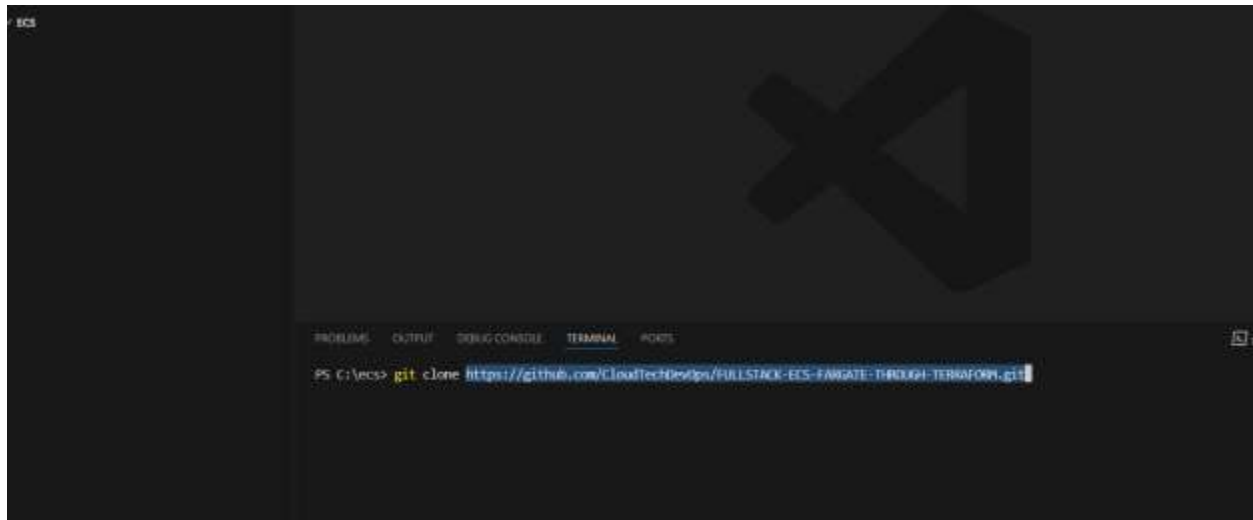
- In your local drive create a folder
- Open that folder In vs code



- Configure the aws credentials in local machine by using aws configure
- Prerequisites for this process aws cli,terraform,vs code must be installed on your local machine
- After that Just clone the ecs terraform repo into your local

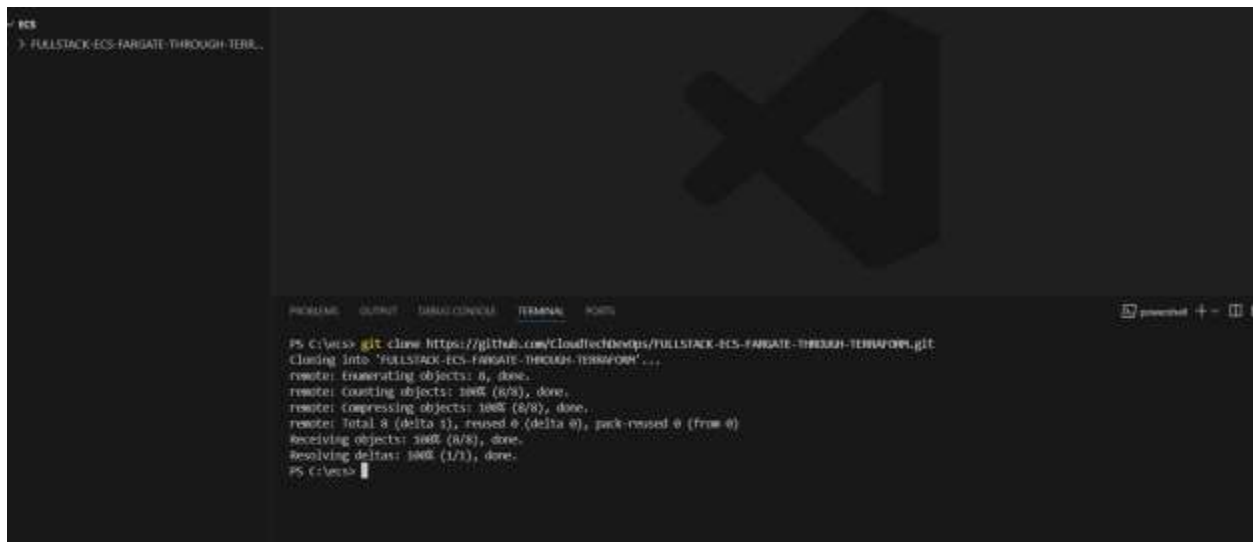
```
git clone https://github.com/CloudTechDevOps/FULLSTACK-ECS-FARGATE-THROUGH-TERRAFORMMM.git
```





A screenshot of a Visual Studio Code terminal window. The terminal shows the command `git clone https://github.com/CloudTechDevs/FULLSTACK-ECS-FARGATE-THROUGH-TERRAFORM.git` being executed. The background of the terminal window features a large, faint logo of a stylized 'X' inside a triangle.

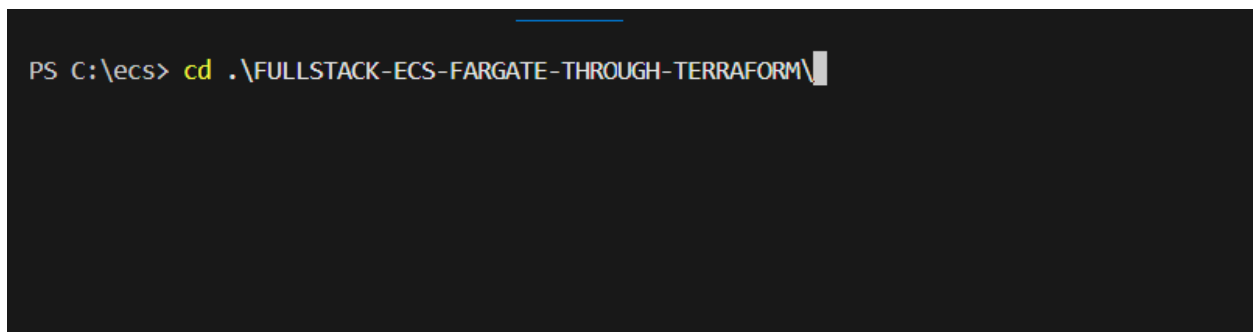
- Cloned the repo successfully



A screenshot of a Visual Studio Code terminal window showing the output of the `git clone` command. The output indicates that the repository was cloned successfully. The background of the terminal window features a large, faint logo of a stylized 'X' inside a triangle.

- Switch to repo

`cd FULLSTACK-ECS-FARGATE-THROUGH-TERRAFORM`



A screenshot of a Visual Studio Code terminal window showing the command `cd .\FULLSTACK-ECS-FARGATE-THROUGH-TERRAFORM\` being executed. The background of the terminal window features a large, faint logo of a stylized 'X' inside a triangle.

```
PROBLEMS  DEBUG CONSOLE  OUTPUT  TERMINAL  PORTS  CODE REFERENCE LOG
FULLSTACK-ECS-FARGATE-THROUGH-TERRAFORM/
H08EY@kiney MITKasa ~/Documents/ECR-REPO
$ cd FULLSTACK-ECS-FARGATE-THROUGH-TERRAFORM/
H08EY@kiney MITKasa ~/Documents/ECR-REPO/FULLSTACK-ECS-FARGATE-THROUGH-TERRAFORM (main)
$ ls
ecs-task/  rds/  vpc-network/
H08EY@kiney MITKasa ~/Documents/ECR-REPO/FULLSTACK-ECS-FARGATE-THROUGH-TERRAFORM (main)
$ cd vpc-network/
H08EY@kiney MITKasa ~/Documents/ECR-REPO/FULLSTACK-ECS-FARGATE-THROUGH-TERRAFORM/vpc-network (main)
```

■ **cd vpc-network**

Run terraform init, terraform plan and terraform apply

Then give cd ..

■ **After change the “rds” Directory.**

Run terraform init, terraform plan and terraform apply – after creating a rds you should launch one manual server

\*install these commands on the server

sudo yum install git -y

sudo yum install mariadb105-server -y

```
(root@ip-172-31-2-231 ~)# yum install git -y
Amazon Linux 2023 Kernel livepatch repository                               136 kB/s | 14 kB   00:00
dependencies resolved.
```

Package	Architecture	Version	Repository	Size
Installing:				
git	x86_64	2.47.1-1.amzn2023.0.2	amazonlinux	54 k
Installing dependencies:				
git-core	x86_64	2.47.1-1.amzn2023.0.2	amazonlinux	4.7 M
git-core-doc	noarch	2.47.1-1.amzn2023.0.2	amazonlinux	2.8 M

```
[ec2-user@ip-172-31-7-58 backend]$ sudo yum install mariadb105-server -y
```

After installing the commands, you need to **git clone** the repository.

Git clone - <https://github.com/CloudTechDevOps/2nd10WeeksofCloudOps-main.git>

```
(root@ip-172-31-2-231 ~)# git clone https://github.com/CloudTechDevOps/2nd10WeeksofCloudOps-main.git
Cloning into '2nd10WeeksofCloudOps-main'...
remote: Enumerating objects: 405, done.
remote: Counting objects: 100% (250/250), done.
remote: Compressing objects: 100% (109/109), done.
remote: Total 405 (delta 213), reused 141 (delta 141), pack-reused 155 (from 1)
Receiving objects: 100% (405/405), 3.66 MiB | 16.28 MiB/s, done.
Resolving deltas: 100% (239/239), done.
[root@ip-172-31-2-231 ~]# ls
2nd10WeeksofCloudOps-main
[root@ip-172-31-2-231 ~]# cd 2nd10WeeksofCloudOps-main/
```

After that switch to the backend directory using the cd command.

```

[root@ip-172-31-2-231 ~]# cd /root/WeeksofCloudOps-main/
[root@ip-172-31-2-231 /root/WeeksofCloudOps-main]# ls
Jenkins-Pipeline-Code  architecture.gif  client  docker-compose.yml  kubernetes-Files  terraform_main_ec2
README.md             backend          docker-compose-pruneas-note.md  eks-terraform  mysql
[root@ip-172-31-2-231 /root/WeeksofCloudOps-main]# cd backend/
[root@ip-172-31-2-231 backend]#

```

After switching run the command mentioned below

```
mysql -h rds-endpoint -u admin -pveeranarni < test.sql
```

- Intilize database in backend directory. Before running the command check your project directory must be in backend directory

Come back to VS code then change the **ecs-task** directory. And add **rds endpoint** near the **db host**.

- Change the ecr image urls in the frontend,backend task files

```

FULLSTACK-ECS-FARGATE-THROUGH-TERRAFORM > backend-task-service.tf > resource "aws_ecs_task_definition" "back-task" > container_definitions
ERR_
32 resource "aws_ecs_task_definition" "back-task" {
33   container_definitions = jsonencode([
34     [
35       {
36         name      = "backend"
37         image      = "545009877819.dkr.ecr.us-east-1.amazonaws.com/backend:latest"
38         cpu        = 256
39         memory     = 512
40         essential = true
41
42         portMappings = [
43           {
44             containerPort = 80
45             protocol      = "tcp"
46           }
47         ]
48
49         environment = [
50           { name = "DB_HOST", value = aws_db_instance.rds.address },
51           { name = "PORT", value = "3306" },
52           { name = "DB_USERNAME", value = "admin" },
53           { name = "DB_PASSWORD", value = "veeranarni" }
54         ]
55       }
56     ]
57   )
58 }
59
60
61
62

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\ecs\FULLSTACK-ECS-FARGATE-THROUGH-TERRAFORM>

- Change the image urls

```
FULLSTACK-ECS-FARGATE-THROUGH-TERRAFORM > frontend-task-service.tf > resource "aws_ecs_task_definition" "front_task" {
  30 }
  31
  32 # ECS Task Definition
  33 resource "aws_ecs_task_definition" "front_task" {
  34   family           = "front-ecs-task"
  35   requires_compatibilities = ["FARGATE"]
  36   network_mode      = "awsvpc"
  37   memory            = "512"
  38   cpu               = "256"
  39   execution_role_arn = aws_iam_role.ecs_task_execution_role.arn
  40
  41   container_definitions = jsonencode([
  42     {
  43       name       = "frontend"
  44       image      = "545009827818.dkr.ecr.us-east-1.amazonaws.com/frontend:latest"
  45       cpu        = 256
  46       memory     = 512
  47       essential = true
  48       portMappings = [
  49         {
  50           containerPort = 80
  51           protocol       = "tcp"
  52         }
  53       ]
  54     }
  55   ])
  56 }
```

■ Give the terraform init

```
PS C:\ecs\FULLSTACK-ECS-FARGATE-THROUGH-TERRAFORM> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.87.0...

```

■ Give the terraform plan

```
PS C:\ecs\FULLSTACK-ECS-FARGATE-THROUGH-TERRAFORM> terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_ecs_task_definition.front_task will be created
```

■ Give the terraform apply

```
PS C:\ecs\FULLSTACK-ECS-FARGATE-THROUGH-TERRAFORM> terraform apply -auto-approve
```

## ■ Resources created

```
aws_db_instance.rds: Still creating... [9m30s elapsed]
aws_db_instance.rds: Still creating... [9m40s elapsed]
aws_db_instance.rds: Still creating... [9m50s elapsed]
aws_db_instance.rds: Still creating... [10m0s elapsed]
aws_db_instance.rds: Still creating... [10m10s elapsed]
aws_db_instance.rds: Still creating... [10m20s elapsed]
aws_db_instance.rds: Still creating... [10m30s elapsed]
aws_db_instance.rds: Still creating... [10m40s elapsed]
aws_db_instance.rds: Still creating... [10m50s elapsed]
aws_db_instance.rds: Still creating... [11m0s elapsed]
aws_db_instance.rds: Still creating... [11m10s elapsed]
aws_db_instance.rds: Still creating... [11m20s elapsed]
aws_db_instance.rds: Still creating... [11m30s elapsed]
aws_db_instance.rds: Still creating... [11m40s elapsed]
aws_db_instance.rds: Still creating... [11m50s elapsed]
aws_db_instance.rds: Still creating... [12m0s elapsed]
aws_db_instance.rds: Still creating... [12m10s elapsed]
aws_db_instance.rds: Still creating... [12m20s elapsed]
aws_db_instance.rds: Still creating... [12m30s elapsed]
aws_db_instance.rds: Still creating... [12m40s elapsed]
aws_db_instance.rds: Still creating... [12m50s elapsed]
aws_db_instance.rds: Creation complete after 12m52s [id=db-x4CF3GG05PARISHQDGI2WH5EI]
aws_ecs_task_definition.back-task: Creating...
aws_ecs_task_definition.back-task: Creation complete after 0s [id=back-end]
aws_ecs_service.back-ecs_service: Creating...
aws_ecs_service.back-ecs_service: Creation complete after 1s [id=arn:aws:ecs:ap-south-1:545089827818:service/my-ecs-cluster/backend-ecs-service]

Apply complete! Resources: 25 added, 0 changed, 0 destroyed.

Outputs:
rds_address = "book-rds.ctatkcwc6r37.ap-south-1.rds.amazonaws.com"
```

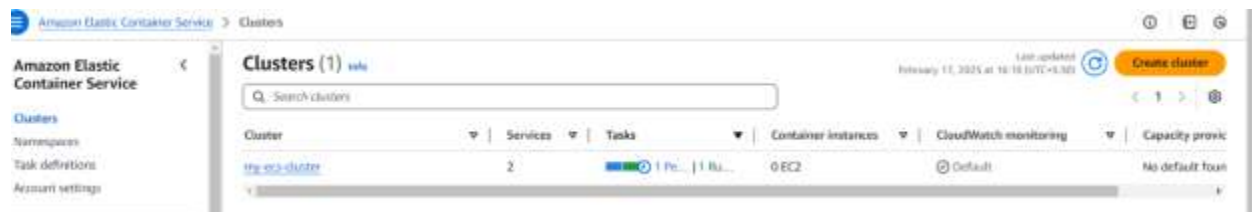
## ■ Open the console and check the load balancers

Load balancers (2)  
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Filter load balancers

<input type="checkbox"/>	Name	DNS name	State	VPC ID	Availability Zones	Type	Date created
<input type="checkbox"/>	backend	backend-1704471441.ap-south-1.elb.amazonaws.com	Active	vpc-010f052c0fa84687	2 Availability Zones	application	February
<input type="checkbox"/>	frontend-alb	frontend-alb-1919077247.ap-south-1.elb.amazonaws.com	Active	vpc-010f052c0fa84687	2 Availability Zones	application	February

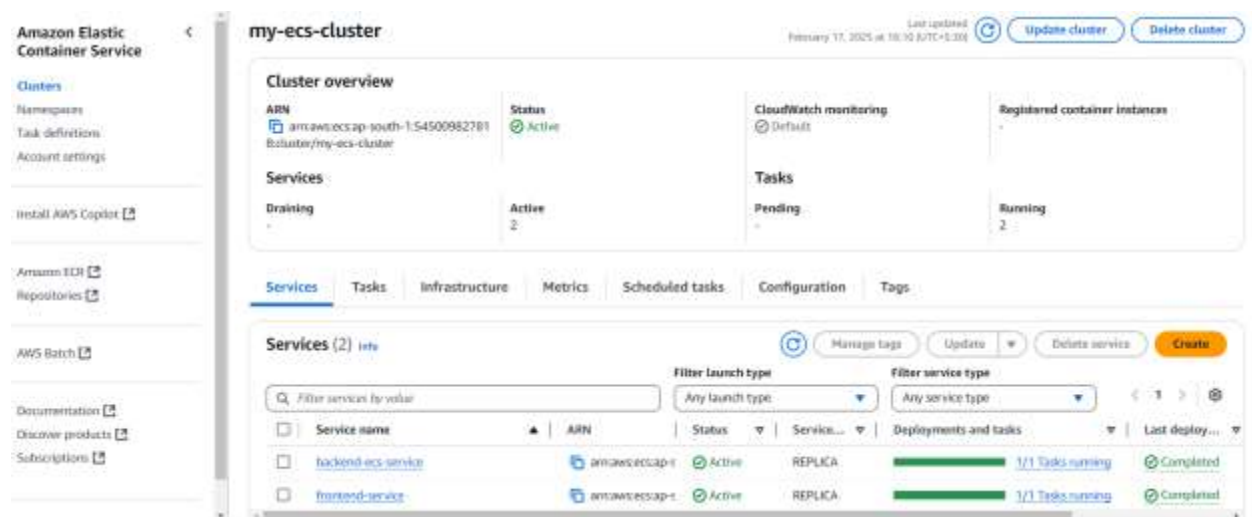
## ■ Open the ecs check the cluster



- check the task definitions



- Open your cluster and check the task status



- Open your domain name
- Click on create record

Public **narni.co.in** [Info](#)

[Delete zone](#) [Test record](#) [Configure query logging](#)

► **Hosted zone details** [Edit hosted zone](#)

**Records (2)** [DNSSEC signing](#) [Hosted zone tags \(0\)](#)

**Records (2)** [Info](#) [Delete record](#) [Import zone file](#) [Create record](#)

Automatic mode is the current search behavior optimized for best filter results. [To change modes go to settings.](#)

Filter records by property or value

Type Routing p... Alias < 1 > ⚙

<input type="checkbox"/>	Record ...	Type	Routin...	Differ...	Alias	Value/Route traffic to	TTL (s...	Health ...	Evalua...
<input type="checkbox"/>	narni.co.in	NS	Simple	-	No	ns-692.awsdns-22.net. ns-1117.awsdns-11.org. ns-1940.awsdns-50.co.uk. ns-176.awsdns-22.com.	172800	-	-
<input type="checkbox"/>	narni.co.in	SOA	Simple	-	No	ns-692.awsdns-22.net. awsd...	900	-	-

- Give sub domain as per your config.js value
- Select alias
- Select application and classic loadbalancer
- Select load balancer region
- Select your backend load balancer
- Create the record

**Quick create record** [Switch to wizard](#)

▼ Record 1 [Delete](#)

Record name [Info](#)  narni.co.in

Record type [Info](#)

Keep blank to create a record for the root domain.

☒ Alias

Route traffic to [Info](#)

Alias to Application and Classic Load Balancer

Asia Pacific (Mumbai)

Alias hosted zone ID: Z9S7NAPLX7UEX

Routing policy [Info](#)

Evaluate target health ☒ Yes

[Add another record](#)

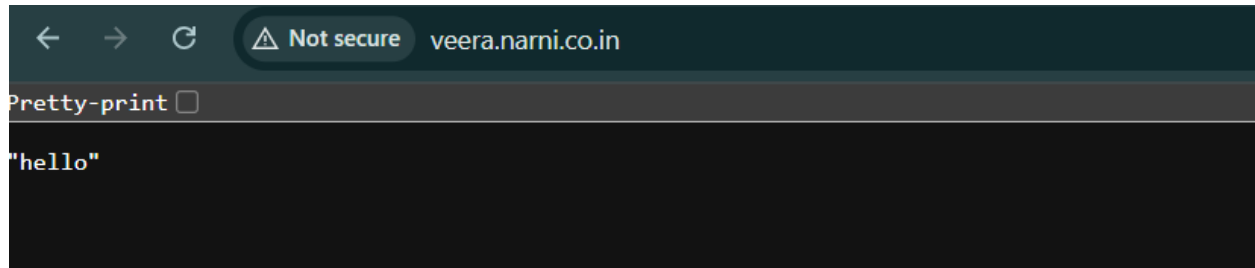
[Cancel](#) [Create records](#)

- Copy the record name

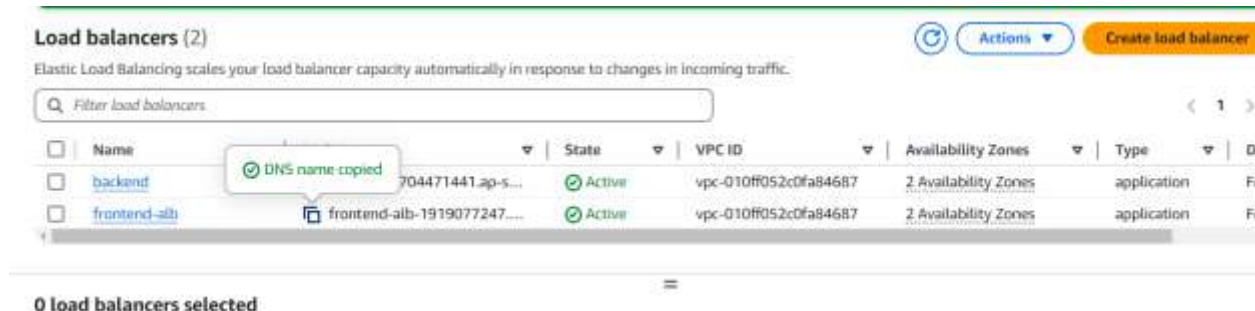
<input type="checkbox"/>	Record name	Type	Routin...	Differ...	Alias	Value/Route traffic to	TTL (s...
<input type="checkbox"/>	narni.co.in	NS	Simple	-	No	ns-692.awsdns-22.net. ns-1117.awsdns-11.org. ns-1940.awsdns-50.co.uk. ns-176.awsdns-22.com.	172800
<input type="checkbox"/>	narni.co.in	SOA	Simple	-	No	ns-692.awsdns-22.net. awsd...	900
<input type="checkbox"/>	veera.narni.co.in	A	Simple	-	Yes	dualstack.backend-1704471...	-



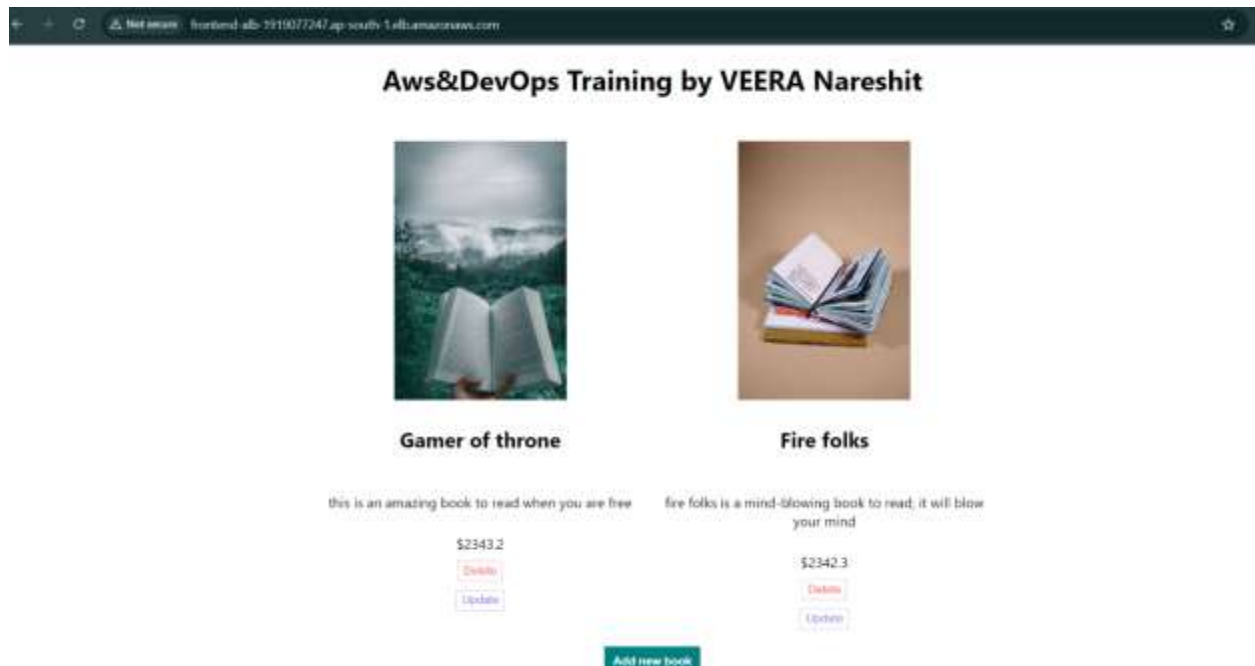
- Search on browser you will get hello response



- Copy the frontend loadblancer url



- Search on web you will get final out put
- Click on add book



-----THANK YOU-----