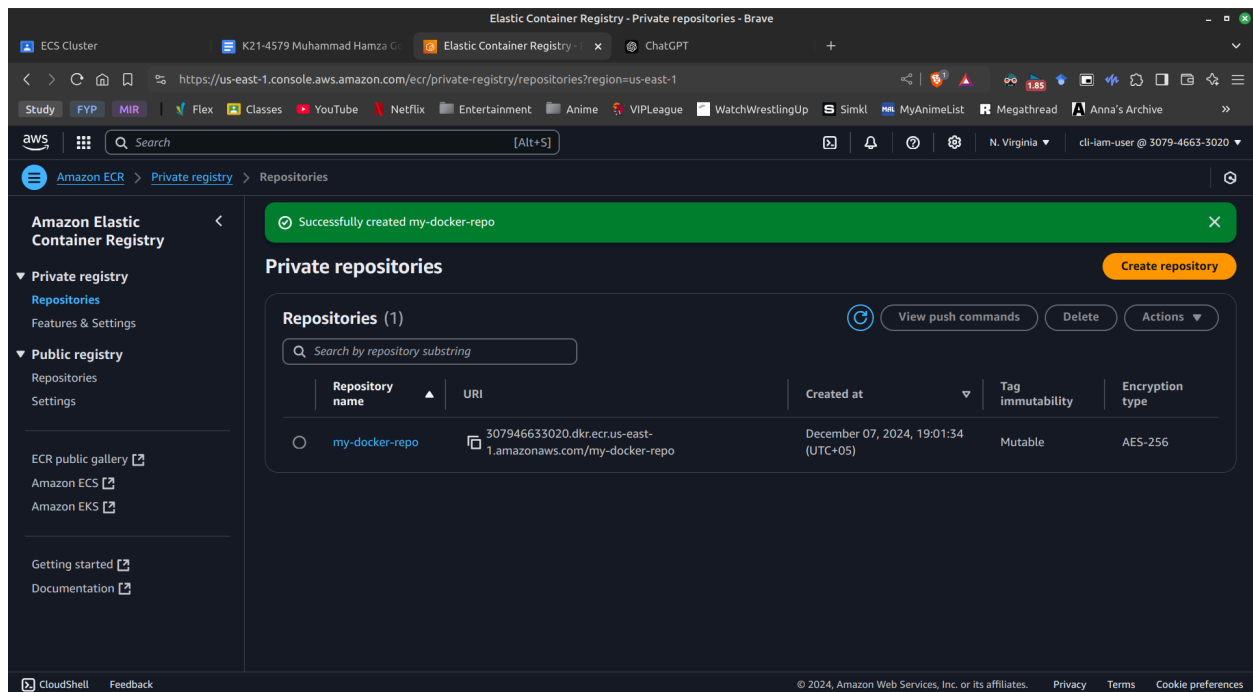


## 1. Create an ECR Repository

1. **Login to AWS Management Console:**
  - Navigate to **ECR (Elastic Container Registry)** from the Services menu.
2. **Create Repository:**
  - Click **Create Repository**.
  - Enter the **repository name** (e.g., **my-docker-repo**).
  - Select **Private** as the repository type.
  - Configure settings like tag immutability or encryption if needed.
  - Click **Create**.



## 2. Push a Docker Image to ECR from Your Local Machine

1. **Authenticate Docker with ECR:**
  - Open your CLI and authenticate

```
aws ecr get-login-password --region us-east-1 | docker login
--username AWS --password-stdin
307946633020.dkr.ecr.us-east-1.amazonaws.com
```

## 2. Tag the Image:

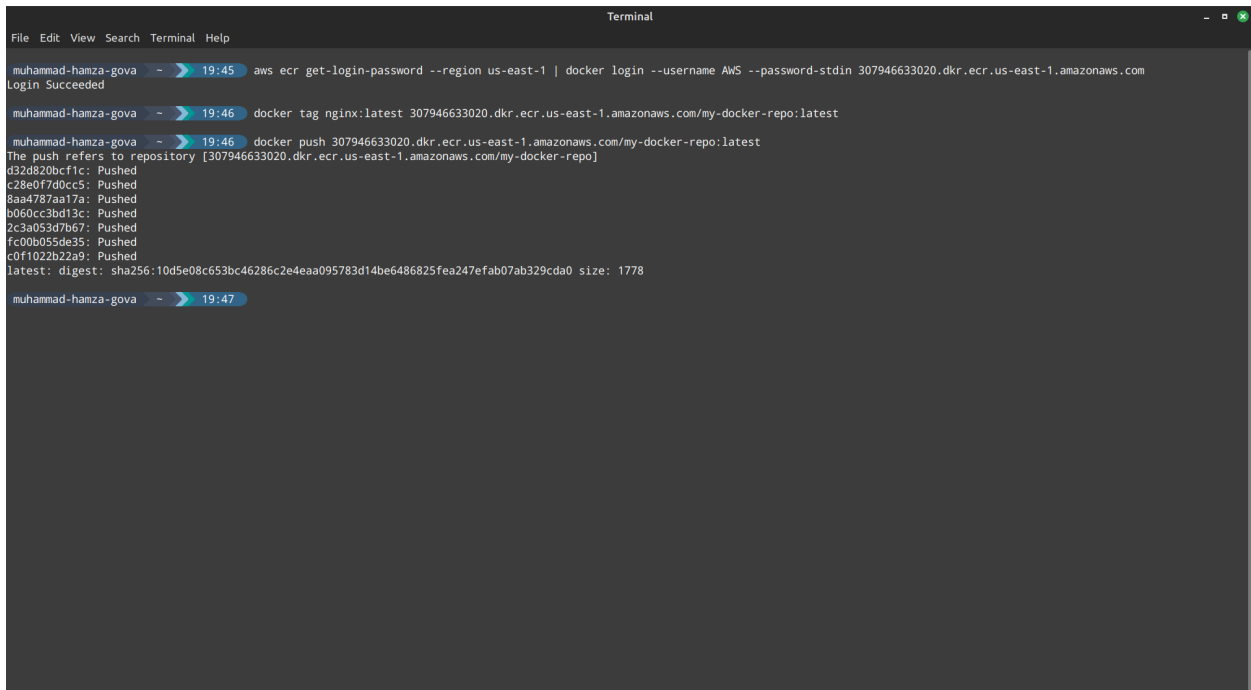
- Tag your Docker image with the repository URI:

```
docker tag nginx:latest  
307946633020.dkr.ecr.us-east-1.amazonaws.com/my-docker-repo:latest
```

## 3. Push the Image:

- Push the tagged image:

```
docker push  
<account-id>.dkr.ecr.<region>.amazonaws.com/<repository-name>:<tag>
```

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows a series of commands and their outputs. The first command is `aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 307946633020.dkr.ecr.us-east-1.amazonaws.com`, which results in "Login Succeeded". The second command is `docker tag nginx:latest 307946633020.dkr.ecr.us-east-1.amazonaws.com/my-docker-repo:latest`. The third command is `docker push 307946633020.dkr.ecr.us-east-1.amazonaws.com/my-docker-repo:latest`, which results in a long list of layers being pushed, including `d32d820bcf1c`, `c28e0f7d0cc5`, `8aa4787aa17a`, `b060cc3bd13e`, `2c3a053d7b67`, `fc00b055de35`, and `c0f1022b22a9`, followed by the `latest` digest and size information.

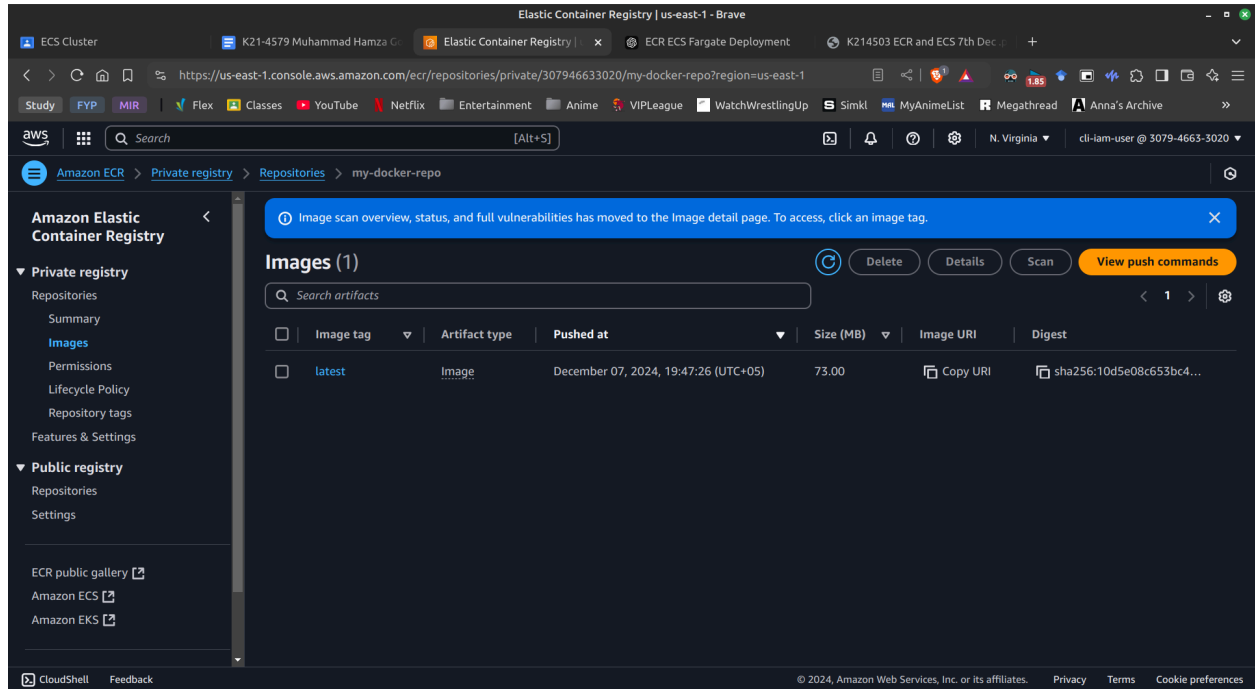
```
muhammad-hamza-gova ~ 19:45 aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 307946633020.dkr.ecr.us-east-1.amazonaws.com
Login Succeeded

muhammad-hamza-gova ~ 19:46 docker tag nginx:latest 307946633020.dkr.ecr.us-east-1.amazonaws.com/my-docker-repo:latest

muhammad-hamza-gova ~ 19:46 docker push 307946633020.dkr.ecr.us-east-1.amazonaws.com/my-docker-repo:latest
The push refers to repository [307946633020.dkr.ecr.us-east-1.amazonaws.com/my-docker-repo]
d32d820bcf1c: Pushed
c28e0f7d0cc5: Pushed
8aa4787aa17a: Pushed
b060cc3bd13e: Pushed
2c3a053d7b67: Pushed
fc00b055de35: Pushed
c0f1022b22a9: Pushed
latest: digest: sha256:10d5e08c653bc46286c2e4eaa095783d14be6486825fea247efab07ab329cda0 size: 1778

muhammad-hamza-gova ~ 19:47
```

Muhammad Hamza  
21K-4579



### 3. Configure Permissions for ECR

#### 1. Create IAM Role for ECR Access:

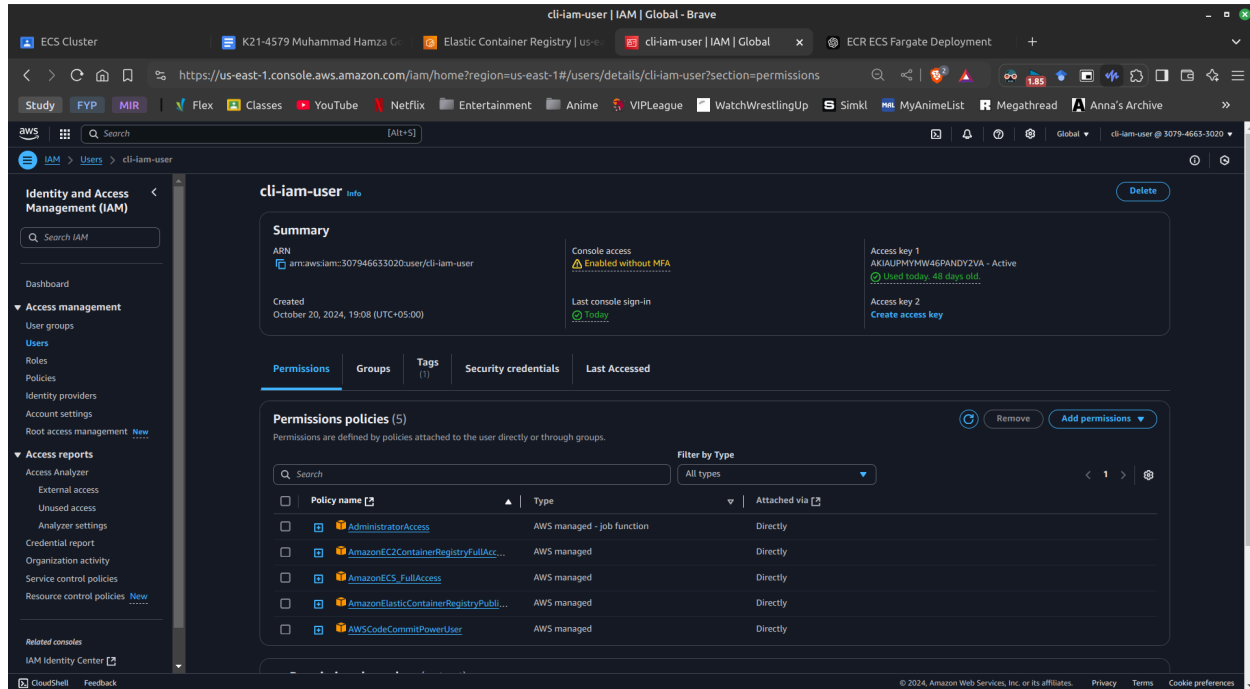
- Go to **IAM** → **Roles** → **Create Role**.
- Select **AWS Service** → **ECS**.
- Attach the policy **AmazonEC2ContainerRegistryFullAccess**.
- Name the role (e.g., **cli-iam-user**) and create it.

#### 2. Assign Policy to User/Role:

- Ensure users pushing/pulling images have the **AmazonEC2ContainerRegistryFullAccess** or similar permissions.

# Muhammad Hamza

## 21K-4579



Here's a step-by-step guide for completing your task based on the AWS Management Console and information from the reference PDF:

## 1. Create an ECR Repository

1. **Login to AWS Management Console:**
  - Navigate to **ECR (Elastic Container Registry)** from the Services menu.
2. **Create Repository:**
  - Click **Create Repository**.
  - Enter the **repository name** (e.g., **my-docker-repo**).
  - Select **Private** as the repository type.
  - Configure settings like tag immutability or encryption if needed.
  - Click **Create**.

## 2. Push a Docker Image to ECR from Your Local Machine

1. **Authenticate Docker with ECR:**

Open your CLI and authenticate:

bash

Copy code

```
aws ecr get-login-password --region <your-region> | docker login
```

```
--username AWS --password-stdin  
<account-id>.dkr.ecr.<region>.amazonaws.com
```

○

## 2. Tag the Image:

Tag your Docker image with the repository URI:

bash

Copy code

```
docker tag <local-image-name>:<tag>  
<account-id>.dkr.ecr.<region>.amazonaws.com/<repository-name>:<tag>
```

○

## 3. Push the Image:

Push the tagged image:

bash

Copy code

```
docker push  
<account-id>.dkr.ecr.<region>.amazonaws.com/<repository-name>:<tag>
```

○

---

## 3. Configure Permissions for ECR

### 1. Create IAM Role for ECR Access:

- Go to **IAM** → **Roles** → **Create Role**.
- Select **AWS Service** → **ECS**.
- Attach the policy **AmazonEC2ContainerRegistryFullAccess**.
- Name the role (e.g., **ECRAccessRole**) and create it.

### 2. Assign Policy to User/Role:

- Ensure users pushing/pulling images have the **AmazonEC2ContainerRegistryPowerUser** or similar permissions.

---

## 4. Set Up an ECS Cluster

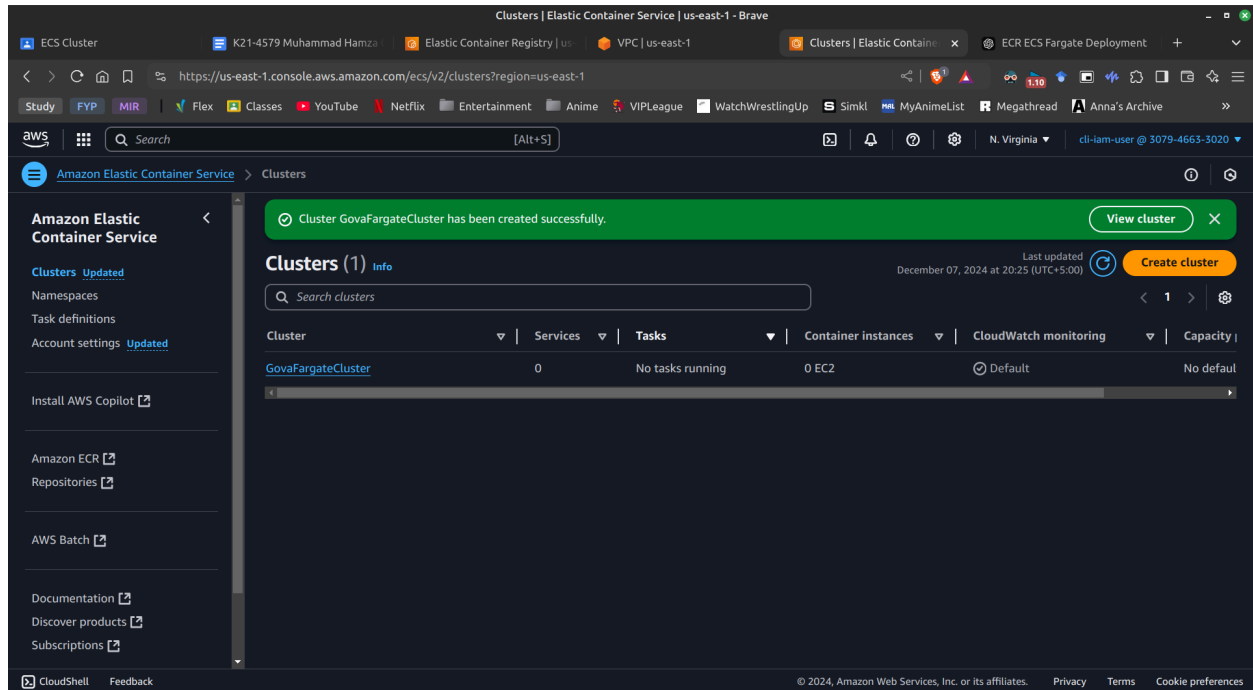
### 1. Create an ECS Cluster:

- Navigate to **ECS (Elastic Container Service)** in the AWS Console.
- Click **Create Cluster**.
- Select **Networking only (Fargate)**.

Muhammad Hamza

21K-4579

- Provide a name for the cluster (e.g., **GovaFargateCluster**).
- Configure VPC and subnets if not pre-configured.
- Click **Create**.



## 5. Deploy a Docker Container to ECS using Fargate

### 1. Create Task Definition:

- Go to **Task Definitions** → **Create New Task Definition**.
- Choose **Fargate**.
- Provide a task definition name (e.g., **MyTaskDefinition**).
- Add a container:
  - Click **Add Container**.
  - Enter the container name (e.g., **MyContainer**).
  - Add the image URI from ECR

**307946633020.dkr.ecr.us-east-1.amazonaws.com/my-docker-repo:latest**

- Configure CPU/memory limits.
- Set port mappings (e.g., 80:80 for a web server).
- Save and create the task definition.

# Muhammad Hamza

## 21K-4579

Task definition containers | Elastic Container Service | us-east-1 - Brave

https://us-east-1.console.aws.amazon.com/ecs/v2/task-definitions/MyTaskDefinition/1/containers?region=us-east-1

Amazon Elastic Container Service

Task definitions > MyTaskDefinition > Revision 1 > Containers

**Task definition successfully created**  
MyTaskDefinition:1 has been successfully created. You can use this task definition to deploy a service or run a task.

**MyTaskDefinition:1**

**Overview**

ARN: arn:aws:ecs:us-east-1:307946633020:task-definition/MyTaskDefinition:1

Status: **ACTIVE**

Time created: December 01, 2024 at 20:31 (UTC+5:00)

App environment: Fargate

Task role: -

Task execution role: ecsTaskExecutionRole

Operating system/Architecture: Linux/x86\_64

Network mode: awsvpc

**Containers**

**Task size**

Task CPU: 512 units (0.5 vCPU)

Task CPU maximum allocation for containers

Task memory: 1,024 MiB (1 GiB)

Task memory maximum allocation for container memory reservation

**Containers**

Container name	Image	Private repository	Essential	CPU	Memory	GPU
MyContainer	3079...	-	Yes	0	-/-	-

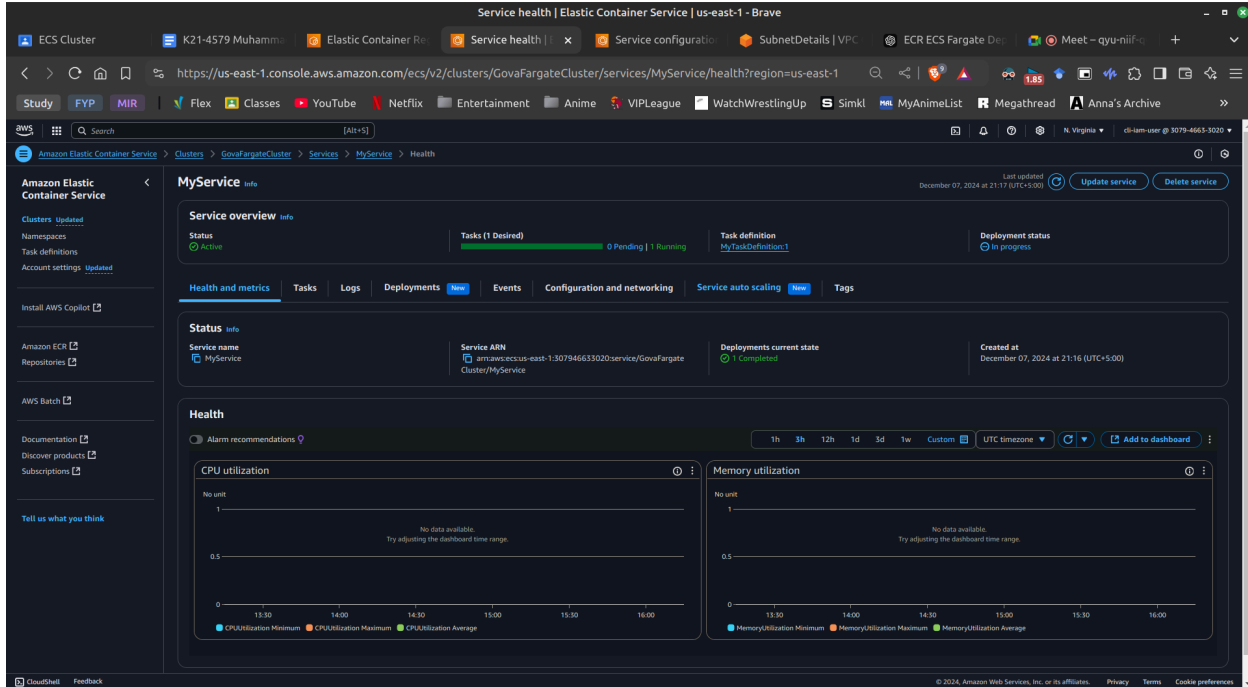
© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

## 2. Run a Service:

- Go to **Services** in your cluster.
- Click **Create Service**.
- Choose Fargate as the launch type.
- Select your task definition and cluster.
- Configure service settings (e.g., number of tasks).
- Enable **Public IP** for networking.
- Click **Create Service**.

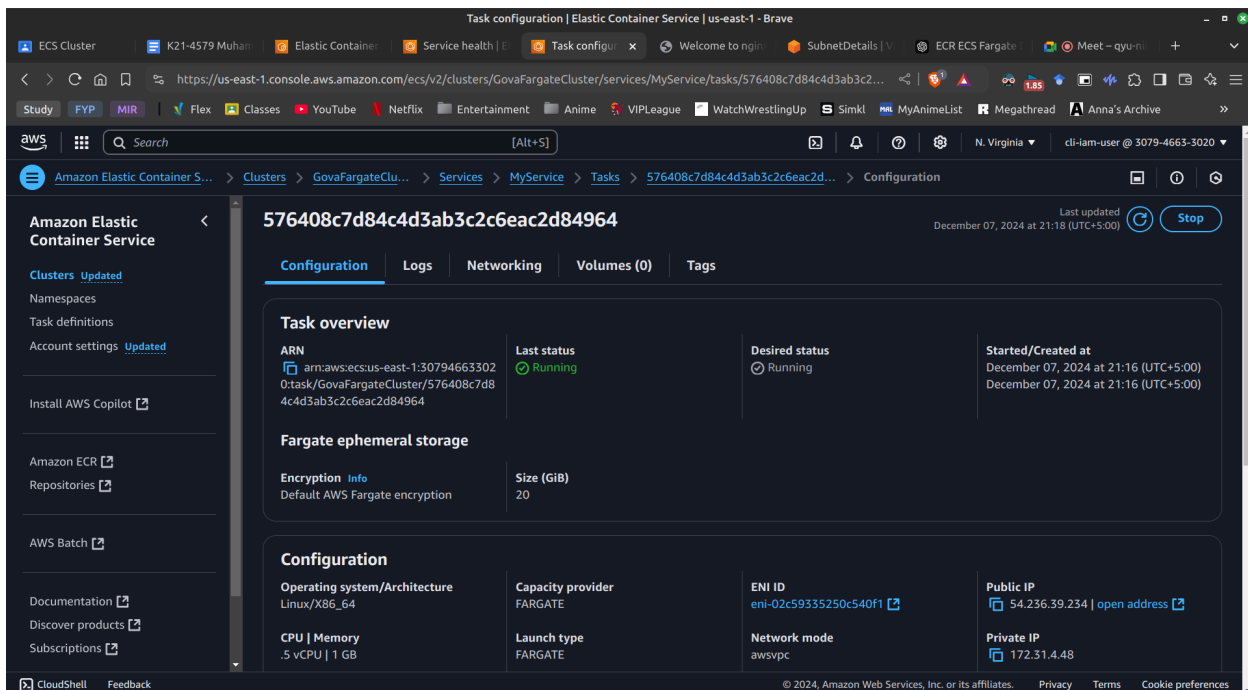
# Muhammad Hamza

## 21K-4579



### 3. Access the Deployed Container:

- Go to the **Tasks** section in your service.
- Copy the **Public IP** of the running task.
- Use it to access your application (e.g., <http://54.236.39.234/>).





# Muhammad Hamza

## 21K-4579

