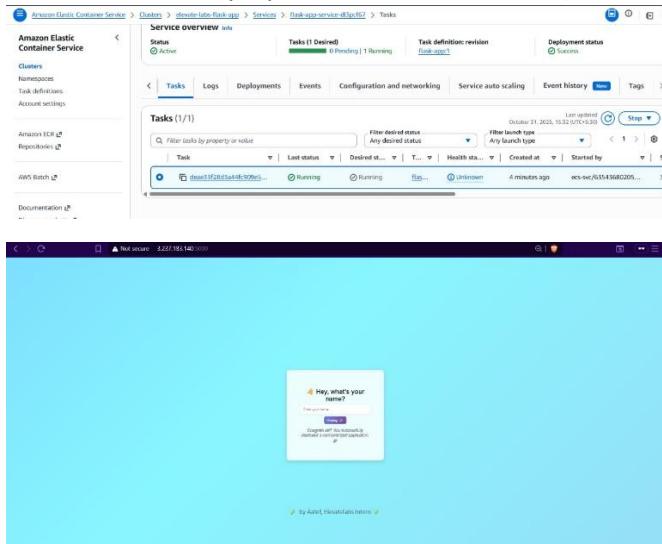


****Deliverables:*** *

* ***Screenshot of running container locally.***

```
[root@localhost elevate-labs-task-8]# docker run -d --name flask-app -p 5000:5000 flask:latest ^C
[root@localhost elevate-labs-task-8]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
 NAMES
d63dfc2615fe      flask:latest       "python app.py"   About a minute ago   Up About a minute   0.0.0.0:5000->5000/tcp, [::]:5000
->5000/tcp          flask-app
[root@localhost elevate-labs-task-8]#
```

* Screenshot of deployed service on the cloud (with URL visible)



* ***Docker file code***

```
# Use official Python image
FROM python:3.12-slim

# Set working directory
WORKDIR /app

# Copy requirements and install
COPY requirements.txt requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Copy app files
COPY app.py .
COPY templates/ ./templates/

# Expose port
EXPOSE 5000

# Run the app
CMD ["python", "app.py"]
```

* ***Short note (4–6 lines) explaining what the container does***

This container runs a simple Flask web application integrated with HTML. It prompts the user for input (like a name) and displays a personalized message. The app is packaged in a Docker image and deployed on AWS ECS using Fargate/EC2. The container exposes port 5000 to handle web requests. Once running, it confirms that the application has been successfully deployed and is accessible via the ECS endpoint.

ELEVATE LABS - TASK #8

Name: Mohammed Aatef

Designation: Cloud Intern

Gmail: moatif1416@gmail.com

GitHub: <https://github.com/aatef14/Elevate-labs-task-8>

Task #8: Deploy a Dockerized Web Application on the Cloud

Objective: To understand *containerization* and *cloud-native deployment* by creating a *Docker image* of a simple web application and deploying it to a *cloud platform*

1. Launch a Amazon Linux 2023 Ec2 machine, connect and download docker.

Steps to install docker in amazon linux (<https://repost.aws/questions/QU1jeKaTRYQ7WeA7XobfP21g/how-do-i-install-docker-version-27-3-1-on-amazon-linux-2023#AN4OjL2dU1T0ONevG7Y3hO-Q>).

2. Once docker installed, run the following cmd to download required files for the flask-app from my git repo.

```
curl -L -o Elevate-labs-task-8.zip https://github.com/aatef14/Elevate-labs-task-8/archive/refs/heads/main.zip
```

```
[root@ip-172-31-28-149 ~]#
[root@ip-172-31-28-149 ~]#
[root@ip-172-31-28-149 ~]# curl -L -o Elevate-labs-task-8.zip https://github.com/aatef14/Elevate-labs-task-8/archive/refs/heads/main.zip
% Total    % Received % Xferd  Average Speed   Time   Time     Time  Current
          Dload  Upload Total Spent   Spent    Left Speed
  0      0      0      0      0       0      0 ---:---:---:---:---:---:---:--- 0
100  3084    0  3084      0      0  14835      0 ---:---:---:---:---:---:--- 14835
[root@ip-172-31-28-149 ~]#
```

3. Then run, **unzip Elevate-labs-task-8.zip**

```
[root@ip-172-31-28-149 ~]# ls
Elevate-labs-task-8.zip
[root@ip-172-31-28-149 ~]# unzip Elevate-labs-task-8.zip
Archive:  Elevate-labs-task-8.zip
10900791ffe6a0e7fb813b773a07ff277d4a05b1
  creating: Elevate-labs-task-8-main/
  creating: Elevate-labs-task-8-main/flask-app-src/
  inflating: Elevate-labs-task-8-main/flask-app-src/Dockerfile
  inflating: Elevate-labs-task-8-main/flask-app-src/app.py
  extracting: Elevate-labs-task-8-main/flask-app-src/requirements.txt
    creating: Elevate-labs-task-8-main/flask-app-src/templates/
    inflating: Elevate-labs-task-8-main/flask-app-src/templates/index.html
[root@ip-172-31-28-149 ~]#
```

4. Then run, cd Elevate-labs-task-8-main/flask-app-src/

```
[root@ip-172-31-28-149 ~]#
[root@ip-172-31-28-149 ~]#
[root@ip-172-31-28-149 ~]#
[root@ip-172-31-28-149 ~]# cd Elevate-labs-task-8-main/flask-app-src/
[root@ip-172-31-28-149 flask-app-src]# ls
Dockerfile app.py requirements.txt templates
[root@ip-172-31-28-149 flask-app-src]#
```

5. Create role for ec2 I have already created a role, steps to create role below.

IAM → Roles → Create role → AWS service → EC2 → Next → Attach policy

AmazonSSMManagedInstanceCore → Next → Name role → Create role

steps to attach role role below.

→ EC2 → Actions → Security → Modify IAM role → Attach the role ✓ .

Add addition permission to the same role. Click on Add permission > attach policy.

The screenshot shows the IAM Roles page with the 'ec2' role selected. The 'Permissions' tab is active, displaying the attached policy 'AmazonSSMManagedInstanceCore'. The ARN of the role is also visible. The left sidebar shows navigation options like Dashboard, Access management, and Access reports.

Select the below mentioned policy, click next and create.

The screenshot shows the 'Other permissions policies' search results page. The policy 'AmazonEC2ContainerRegistryFullAccess' is selected. Other policies listed include AmazonEC2ContainerRegistryPowerUser, AmazonEC2ContainerRegistryPullOnly, AmazonEC2ContainerRegistryReadOnly, AmazonEC2ContainerServiceAutoscaleRole, and AmazonEC2ContainerServiceEventsRole. The page includes a search bar, filter by type, and a table with columns for Policy name, Type, and Description.

10. Go to ECR

The screenshot shows the AWS Lambda search results page. A search bar at the top contains the text "ecr". Below the search bar, there is a sidebar with "EC2" selected. The main content area is titled "Services" and lists several services: "Elastic Container Registry" (selected), "Secrets Manager", and "Key Management Service". Each service entry includes a brief description and a "Show more" link. The "Elastic Container Registry" entry also has a "Top features" section with "Repositories" and "Private registry".

11. Click on private repositories and click “Click Repositories”.

The screenshot shows the "Private repositories" page under the "Private registry" section of the Amazon ECR console. The left sidebar shows "Private registry" selected. The main area displays a table with columns for "Repository name", "URI", "Created at", "Tag immutability", and "Encryption type". A message at the bottom states "No repositories" and "No repositories were found". Action buttons include "View push commands", "Delete", "Actions", and "Create repository" (which is highlighted with an orange arrow).

12. Name the repo example = below screenshot, leave everything default, scroll down, and click create.

The screenshot shows the "Create private repository" form. The left sidebar shows "Private registry" selected. The main form has a "General settings" tab active. Under "Repository name", the input field contains "644118594096.dkr.ecr.us-east-1.amazonaws.com/elevate-labs-repo/flask-app". The "Image tag settings" tab is also visible, containing sections for "Image tag mutability" (with "Mutable" selected) and "Mutable tag exclusions".

13. Once created click on that repo

The screenshot shows the Amazon ECR console under the 'Private registry' section. A green success message at the top states 'Successfully created elevate-labs-repo/flask-app'. Below it, a table lists a single private repository: 'elevate-labs-repo/flask-app' with URI '644118594096.dkr.ecr.us-east-1.amazonaws.com/elevate-labs-repo/flask-app'. The table includes columns for Repository name, URI, Created at, Tag immutability, and Encryption type.

14. Click on "view push command"

The screenshot shows the 'Images' page for the 'elevate-labs-repo/flask-app' repository. It displays a message: 'Image scan overview, status, and full vulnerabilities has moved to the Image detail page. To access, click an image tag.' Below this, there is a table with no images listed, showing columns for Image tag, Artifact type, Pushed at, Size (MB), Image URI, Digest, and Last recorded pull time.

15. Copy paste give command one by one. Eg

First command.

The screenshot shows a modal window titled 'Push commands for elevate-labs-repo/flask-app'. It has tabs for 'macOS / Linux' (selected) and 'Windows'. A note at the top says: 'Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see [Getting Started with Amazon ECR](#) [2].'

Step 1: '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](#) [2].'

A callout bubble indicates 'Code copied' for the first command:

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 644118594096.dkr.ecr.us-east-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.

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

```
docker build -t elevate-labs-repo/flask-app .
```

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

```
docker tag elevate-labs-repo/flask-app:latest 644118594096.dkr.ecr.us-east-1.amazonaws.com/elevate-labs-repo/flask-app:latest
```

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

```
docker push 644118594096.dkr.ecr.us-east-1.amazonaws.com/elevate-labs-repo/flask-app:latest
```

At the bottom right of the modal is a 'Close' button.

First command in terminal

```
[root@ip-172-31-28-149 flask-app-src]# aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 644118594096.dkr.ecr.us-east-1.amazonaws.com
WARNING! Your credentials are stored unencrypted in '/root/.docker/config.json'.
Configure a credential helper to remove this warning. See
https://docs.docker.com/go/credential-store/
Login Succeeded
[root@ip-172-31-28-149 flask-app-src]#
```

Second command

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 elevate-labs-repo/flask-app .`

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

`docker tag elevate-labs-repo/flask-app:latest 644118594096.dkr.ecr.us-east-1.amazonaws.com/elevate-labs-repo/flask-app:latest`

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

`docker push 644118594096.dkr.ecr.us-east-1.amazonaws.com/elevate-labs-repo/flask-app:latest`

[Close](#)

Second command in terminal

```
[root@ip-172-31-28-149 flask-app-src]# docker build -t elevate-labs-repo/flask-app .
[+] Building 0.3s (11/11) FINISHED
-> [internal] load build definition from Dockerfile
-> => transferring dockerfile: 434B
-> [internal] load metadata for docker.io/library/python:3.12-slim
-> [internal] load .dockerignore
-> => transferring context: 2B
-> [1/6] FROM docker.io/library/python:3.12-slim@sha256:e97cf9a2e84d604941d9902f00616db7466ff302af4b1c3c67fb7c522efa8ed9
-> [internal] load build context
-> => transferring context: 356B
-> CACHED [2/6] WORKDIR /app
-> CACHED [3/6] COPY requirements.txt requirements.txt
-> CACHED [4/6] RUN pip install --no-cache-dir -r requirements.txt
-> CACHED [5/6] COPY app.py .
-> CACHED [6/6] COPY templates/ ./templates/
-> exporting to image
-> => exporting layers
-> => writing image sha256:4e98ebc3ea8a016eeb9d5993f6aa5a52978379d907ca50e7dde4d49936d67e0
-> => naming to docker.io/elevate-labs-repo/flask-app
[root@ip-172-31-28-149 flask-app-src]#
```

Third Command

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

`docker tag elevate-labs-repo/flask-app:latest 644118594096.dkr.ecr.us-east-1.amazonaws.com/elevate-labs-repo/flask-app:latest`

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

`docker push 644118594096.dkr.ecr.us-east-1.amazonaws.com/elevate-labs-repo/flask-app:latest`

[Close](#)

Third command in terminal

```
[root@ip-172-31-28-149 flask-app-src]#
[root@ip-172-31-28-149 flask-app-src]#
[root@ip-172-31-28-149 flask-app-src]# docker tag elevate-labs-repo/flask-app:latest 644118594096.dkr.ecr.us-east-1.amazonaws.com/elevate-labs-repo/flask-app:latest
[root@ip-172-31-28-149 flask-app-src]# docker images
REPOSITORY          TAG      IMAGE ID   CREATED        SIZE
644118594096.dkr.ecr.us-east-1.amazonaws.com/elevate-labs-repo/flask-app    latest    4e98ebc3ea8a  18 minutes ago  132MB
elevate-labs-repo/flask-app                                         latest    4e98ebc3ea8a  18 minutes ago  132MB
flask-app                                         latest    4e98ebc3ea8a  18 minutes ago  132MB
[root@ip-172-31-28-149 flask-app-src]#
```

Fourth command

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

```
 docker push 644118594096.dkr.ecr.us-east-1.amazonaws.com/elevate-labs-repo/flask-app:latest
```

[Close](#)

Fourth command in terminal

```
[root@ip-172-31-28-149 flask-app-src]#
[root@ip-172-31-28-149 flask-app-src]#
[root@ip-172-31-28-149 flask-app-src]# docker push 644118594096.dkr.ecr.us-east-1.amazonaws.com/elevate-labs-repo/flask-app:latest
The push refers to repository [644118594096.dkr.ecr.us-east-1.amazonaws.com/elevate-labs-repo/flask-app]
8f929654a514: Pushed
619c1b814cd6: Pushed
229827332deb: Pushed
f08e023fb09c: Pushed
9d8dfb9954dd: Pushed
7770fbba416af: Pushed
acbbee3380a1: Pushed
16dffb6a6636: Pushed
d7c97cb6f1fe: Pushed
latest: digest: sha256:7ee3eee85203f83eee1624054a1133094c4354656fb6947d501c26578036da73 size: 2198
[root@ip-172-31-28-149 flask-app-src]#
```

Images successfully pushed to the private repositories we created.

The screenshot shows the Amazon ECR Private registry interface. On the left, there's a sidebar with navigation links for 'Amazon Elastic Container Registry', 'Private registry' (which is expanded), and 'Public registry'. Under 'Private registry', there are links for 'Repositories', 'Summary', 'Images' (which is selected and highlighted in blue), 'Permissions', 'Lifecycle Policy', 'Repository tags', and 'Features & Settings'. The main content area has a green header bar with the text 'Images (1)'. Below the header is a search bar labeled 'Search artifacts'. A table lists the pushed image details:

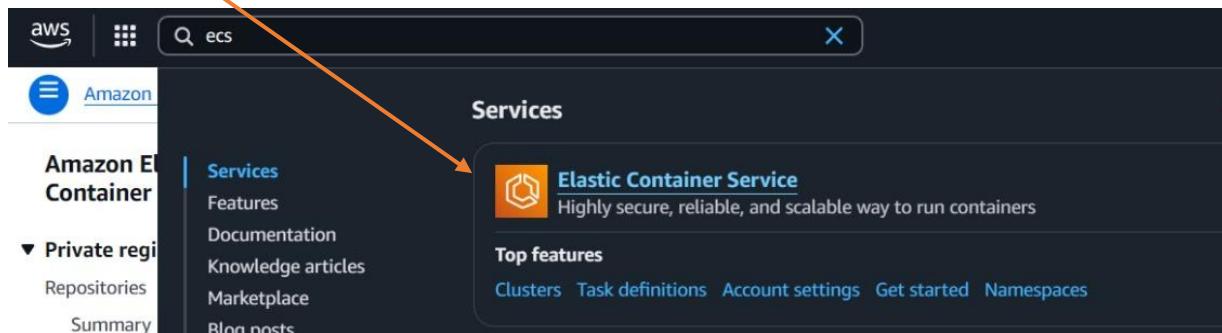
Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Last recorded pull time
latest	Image	October 31, 2025, 15:05:57 (UTC+05:5)	49.11	<input type="button" value="Copy URI"/>	<input type="button" value="sha256:7ee3eee85203f83..."/>	-

At the top right of the main area, there are buttons for 'Delete', 'Details', 'Scan', and 'View push commands'.

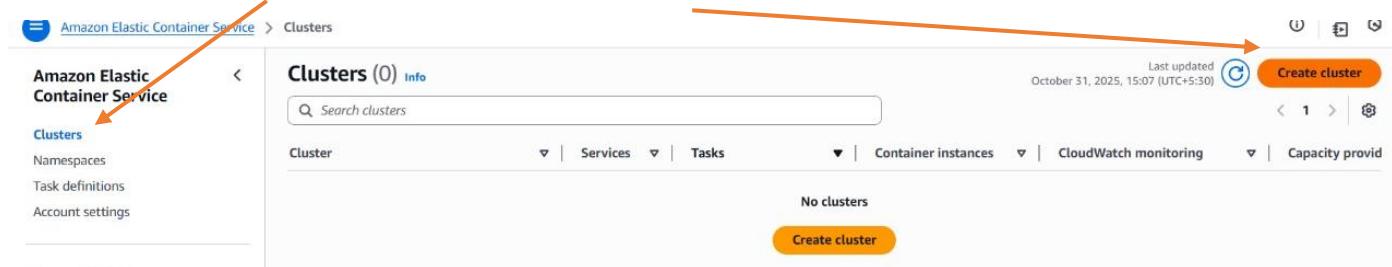
16. Create one more role step below

IAM > ROLES > Create Roles > select AWS service > use case Elastic Container service > click next > name the role > click create.

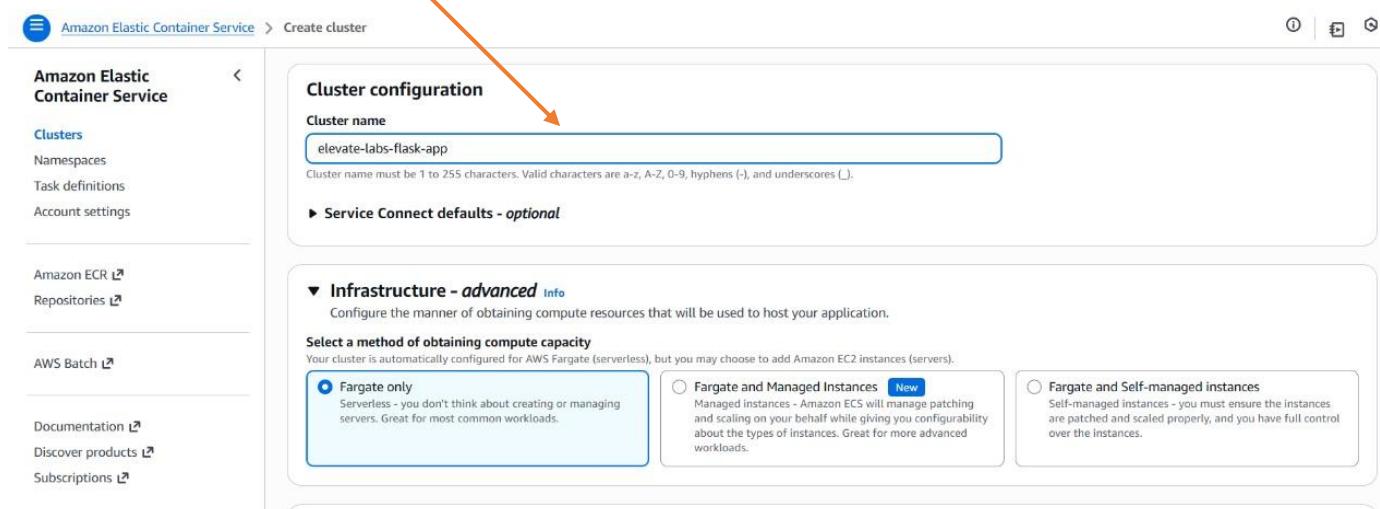
17. Go to ECS.



18. Select clusters in left pane and click "Create Cluster".



19. Name the cluster, leave everything default and simply create.



20. Go to task definition in left pane and click “Create New Task Definition”

Amazon Elastic Container Service

Clusters Namespaces Task definitions Account settings

Amazon ECR Repositories

Task definitions (0) Last updated: October 31, 2025, 15:14 (UTC+5:30) Deploy Create new revision Create new task definition

Filter task definitions Active Status of last revision

Create new task definition Create new task definition with JSON

21. Name it and select “AWS fargate”

Amazon Elastic Container Service > Create new task definition

Amazon Elastic Container Service

Clusters Namespaces Task definitions Account settings

Amazon ECR Repositories

AWS Batch Documentation Discover products Subscriptions

Create new task definition

Task definition configuration

Task definition family: flask-app

Infrastructure requirements

Launch type: AWS Fargate

Managed Instances - new

Amazon EC2 instances

22. leave task role blank and select “Create new role” in task execution role.

Amazon Elastic Container Service > Create new task definition

Amazon Elastic Container Service

Clusters Namespaces Task definitions Account settings

Amazon ECR Repositories

AWS Batch

Create new task definition

Task roles - conditional

Task role: -

Task execution role: Create new role

Task placement - optional

Fault injection - optional

More steps below

23. In container select “Browse ECR images”

Amazon Elastic Container Service > Create new task definition

Container - 1 Info

Container details
Specify a name, container image, and whether the container should be marked as essential. Each task definition must have at least one essential container.

Name flask-app **Essential container** Yes

Image URI repository-uri/image:tag **Browse ECR images**

Private registry Info
Store credentials in Secrets Manager, and then use the credentials to reference images in private registries.

Private registry authentication

Port mappings Info
Add port mappings to allow the container to access ports on the host to send or receive traffic. For port name, a default will be assigned if left blank.

Container port	Protocol	Port name	App protocol
5000	TCP	container-port-protocol	HTTP

Add port mapping

24. Select the private repo we created in step 10, and select the image

Select Amazon ECR image

Private repository
Select the repository containing the image you want to use.

Choose a repository ✖

elevate-labs-repo/flask-app
644118594096.dkr.ecr.us-east-1.amazonaws.com/elevate-labs-repo/flask-app elevate-labs-repo/flask-app

End of all private ECR repositories ⚙️

Image tag **Image digest** **Pushed at**

No images

Select image by

Image digest
Use the SHA256 digest to reference this image.

Image tag
Select an image to use tags.

Cancel **Select image**

Find images

Image tag **Image digest** **Pushed at**

latest sha256:7ee3eee85203f83eee1624... October 31, 2025, 15:05 (UTC+5:30)

Select image by

Image digest
Use the SHA256 digest to reference this image.

Image tag
Use a human-readable tag to reference this image.

Cancel **Select image**

25. In environment variables, keep the key as it is and value can be anything.

Amazon Elastic Container Service > Create new task definition

Environment variables | [Info](#)

Add individually
Add a key-value pair to specify an environment variable.

Key	Value type	Value
FOOTER_TEXT	Value	iatef, ElevateLabs Intern

[Remove](#)

[Add environment variable](#)

Add from file
Add environment variables in bulk by providing an environment file hosted on Amazon S3.

[Add environment file](#)

You can add 10 more environment files.

▼ Logging - optional

26. Leave everything default and click create.

Amazon Elastic Container Service > Create new task definition

Container network settings - optional

Docker configuration - optional

Resource limits (Ulimits) - optional

Docker labels - optional

[+ Add container](#)

Storage - optional

Monitoring - optional
Configure your application trace and metric collection settings using the AWS Distro for OpenTelemetry integration.

Tags - optional | [Info](#)
Tags help you to identify and organize your task definitions.

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

27. Navigate to clusters in left pane and click on your cluster.

Amazon Elastic Container Service > Clusters

Clusters (1) | [Info](#)

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

Notifications 0 0 2 0 0 0

Last updated October 31, 2025, 15:23 (UTC+5:30) [Edit](#) [Create cluster](#)

Cluster	Services	Tasks	Container instances	CloudWatch monitoring	Capacity provider
elevate-labs-flask-app	0	No tasks running	0 EC2	Default	No default found

28. scroll down and in service tab click "create".

Amazon Elastic Container Service > Clusters > elevate-labs-flask-app > Services

Cluster overview

ARN	Status	CloudWatch monitoring	Registered container instances
arn:aws:ecs:us-east-1:644118594096:cluster/elevate-labs-flask-app	Active	Default	-

Services

Draining	Active	Pending	Running
-	1	-	1

Tasks

Draining	Active	Pending	Running
-	1	-	1

[Services](#) [Tasks](#) [Infrastructure](#) [Updated](#) [Metrics](#) [Scheduled tasks](#) [Configuration](#) [Updated](#) [Event history](#) [New](#) [Tags](#)

Services (1) | [Info](#)

Last updated October 31, 2025, 15:29 (UTC+5:30) [Edit](#) [Manage tags](#) [Update](#) [Delete service](#) [Create](#)

29. In task definition family select the task definition we created in step 20 and name the service.

Amazon Elastic Container Service > Clusters > elevate-labs-flask-app > Create service

Cluster elevate-labs-flask-app has been created successfully. View cluster

Create service Info

Service details

Task definition family
Select an existing task definition family. To create a new task definition, go to Task definitions ↗

Select a task definition family flask-app (C)

Filter task definitions by family name prefix flask-app (C)

End of all task definition families flask-app (C)

Service name
Assign a service name that is unique for this cluster.

Up to 255 letters (uppercase and lowercase), numbers, underscores, and hyphens are allowed. Service names must be unique within a cluster.

Environment AWS Fargate

Existing cluster elevate-labs-flask-app

Tell us what you think

30. In networking keep everything default.

Amazon Elastic Container Service > Clusters > elevate-labs-flask-app > Create service

Networking

VPC Info
Select a VPC to use for your Amazon ECS resources.
vpc-0fac7e34b582cecf4 (C) Create a new VPC ↗

Subnets
Choose the subnets within the VPC that the task scheduler should consider for placement.
Choose subnets (C) Clear current selection

subnet-0af47a9d68c0c8ef9 X us-east-1b 172.31.0.0/20
subnet-0307332ee05010c7 X us-east-1e 172.31.48.0/20
subnet-045adcab200049a54 X us-east-1d 172.31.16.0/20
subnet-0f5dc30ab1d45256a X us-east-1f 172.31.64.0/20
subnet-03e8868798b6bc583 X us-east-1c 172.31.80.0/20
subnet-0af948487674e101a X us-east-1a 172.31.32.0/20

31. next create ne security group NOT default, add rule to allow traffic from everywhere for port 5000 like below, rest keep everything default and click create.

Amazon Elastic Container Service > Clusters > elevate-labs-flask-app > Create service

Security group Info
Choose an existing security group or create a new security group.
 Use an existing security group
 Create a new security group

Security group details
Specify the configuration to use when creating the new security group.

Security group name ecs-cf2h3be

Security group description Created in ECS Console

Security group description must be 1 to 255 characters. Valid characters are a-z, A-Z, 0-9, underscores (_), hyphens (-), colons (:), forward slashes (/), parentheses (), hashtags (#), commas (,), at signs (@), brackets ([]), plus signs (+), equal signs (=), ampersands (&), semicolons (;), brackets ([]), exclamation points (!), dollar signs (\$), asterisks (*).

Inbound rules for security groups
Add one or more ingress rules for your security group.

Type	Protocol	Port range	Source	Values	Delete
Custom TCP	TCP	5000	Custom	0.0.0.0/0	Delete

Enter a valid port or port range between 0 and 65535. For example: 80 or 0-1023.

Add rule

Public IP Info
Choose whether to auto-assign a public IP to the task's elastic network interface (ENI).
 Turned on

Keep public IP enabled.

32. Once service is created click on service.

Amazon Elastic Container Service < Clusters > elevate-labs-flask-app > Services

Cluster overview

ARN arn:aws:ecs:us-east-1:644118594096:cluster/elevate-labs-flask-app Status Active CloudWatch monitoring Default Registered container instances -

Services

Draining Active Pending Running 1

Tasks

Services (1) Info Last updated October 31, 2025, 15:29 (UTC+5:30)

Service name ARN Status Schedul... Lau... Task de... Deployments and tasks

flask-app-service-dl3pcf67 arn:aws:ecs:us-e Active REPLICA flask-app:1 1/1 Tasks

Services Tasks Infrastructure Updated Metrics Scheduled tasks Configuration Updated Event history New Tags

Services (1) Info Last updated October 31, 2025, 15:29 (UTC+5:30)

Filter services by value Manage tags Update Delete service Create

Filter launch type Any launch type Filter scheduling strategy Any scheduling strategy

1 1 1

Tell us what you think

33. Then scroll down and click on tasks.

Amazon Elastic Container Service < Clusters > elevate-labs-flask-app > Services > flask-app-service-dl3pcf67 > Tasks

Service overview Info

Status Active Tasks (1 Desired) 0 Pending | 1 Running Task definition: revision flask-app:1 Deployment status Success

Tasks (1/1) Last updated October 31, 2025, 15:32 (UTC+5:30)

Filter tasks by property or value Filter desired status Any desired status Filter launch type Any launch type

Task Last status Desired st... T... Health sta... Created at Started by Sta

deae33f28d3a44fc909e5... Running Running flas... Unknown 4 minutes ago ecs-svc/63543680205... 3 m

Tasks Logs Deployments Events Configuration and networking Service auto scaling Event history New Tags

34. Scroll down and you'll get a public IP

Amazon Elastic Container Service < ... > Services > flask-app-service-dl3pcf67 > Tasks > deae33f28d3a44fc909e50b528614485 > Configuration

Fargate ephemeral storage

Encryption Info Default AWS Fargate encryption Size (GiB) 20

Configuration

Operating system/Architecture Linux/X86_64 Capacity provider Fargate ENI ID eni-057b2e2e3b54a97fb

CPU | Memory 1 vCPU | 2 GB Launch type Fargate Network mode awsvpc

Platform version 1.4.0 Task definition: revision flask-app:1 Subnet subnet-0af47a9d68c0c8ef9

Task execution role Task group

Container details for flask-app

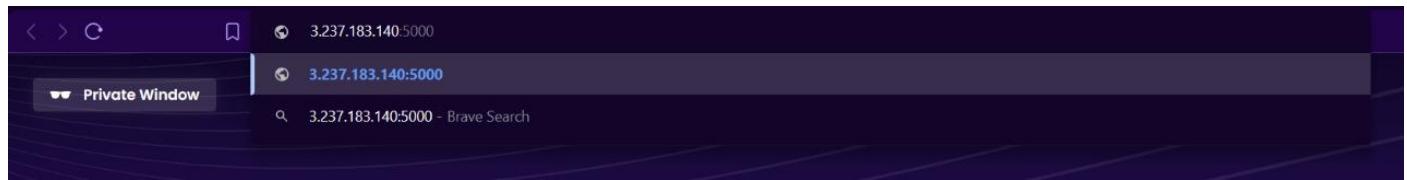
Details Log configuration Restart policy Network bindings Docker labels and hosts Environment variables and files

Public IP 3.237.183.140 | open address

Private IP 172.31.5.153

MAC address 02:e5:77:bd:cb:db

35. Paste the ip like this in browser.



36. The app will open like below with custom footer set during environment variable setting.

