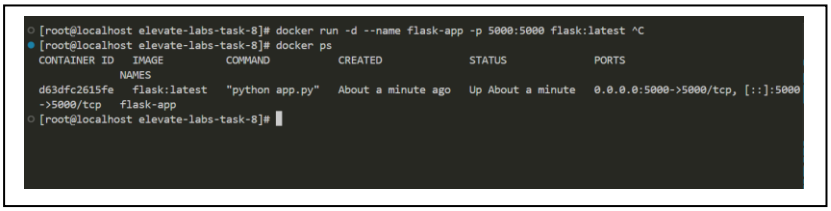
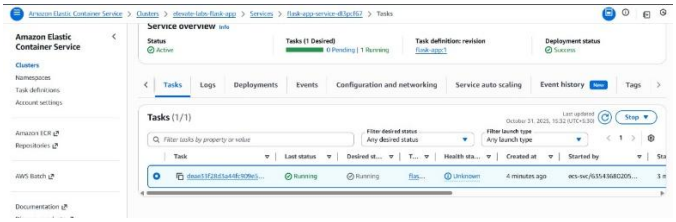
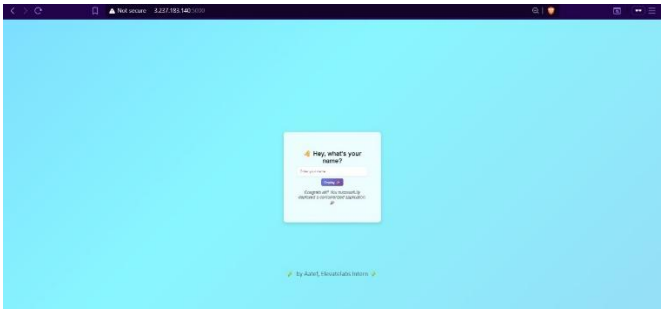
📦 ***Deliverables: ***

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



* 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#AN4OiL2dU1T0ONevG7Y3hO-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    Left  Speed
  0     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]#
```

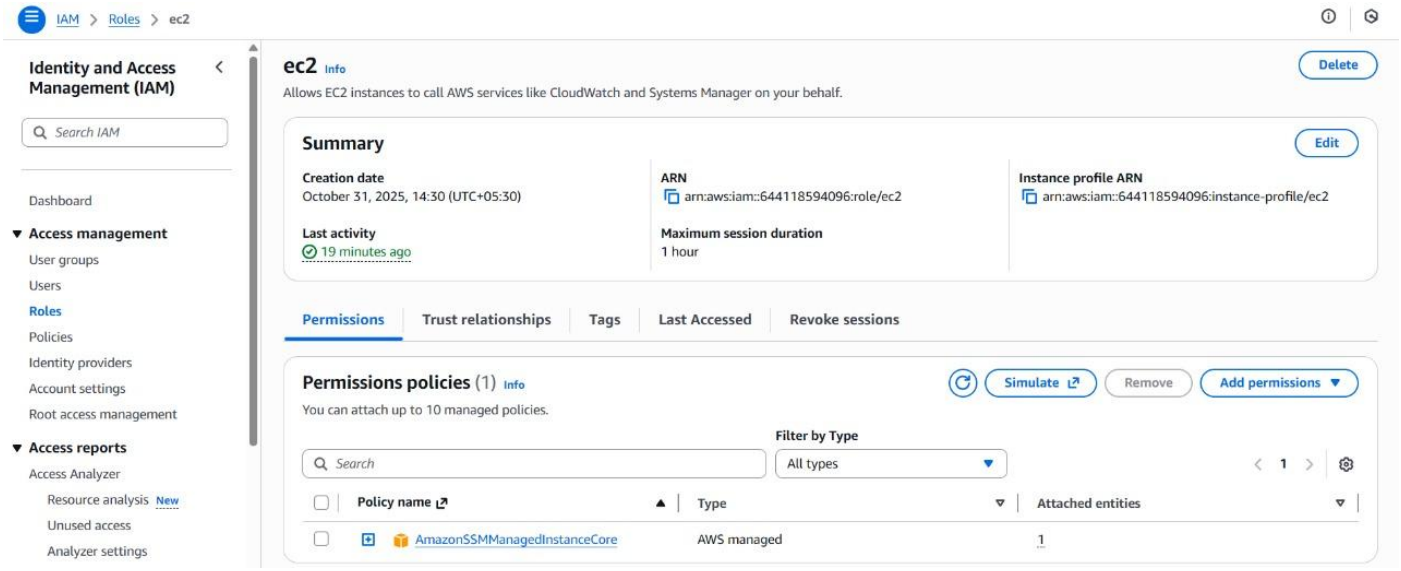
5. Create role for ec2 I have already created a role, steps to create role below.

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

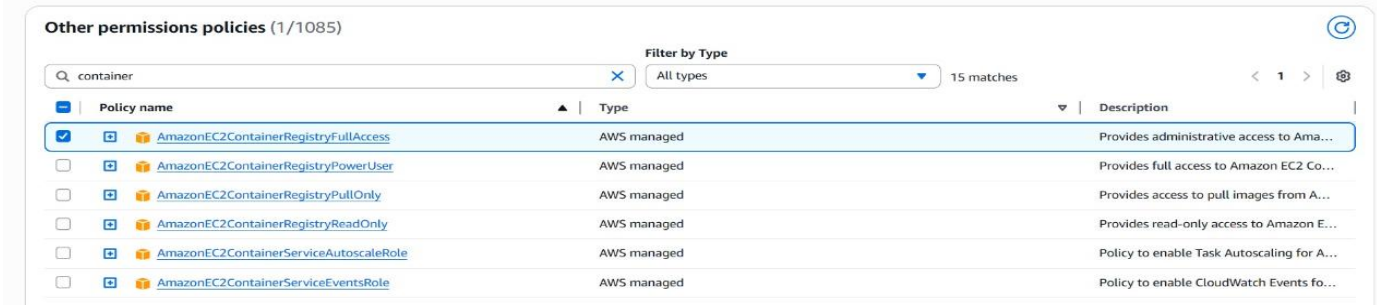
steps to attach role role below.

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

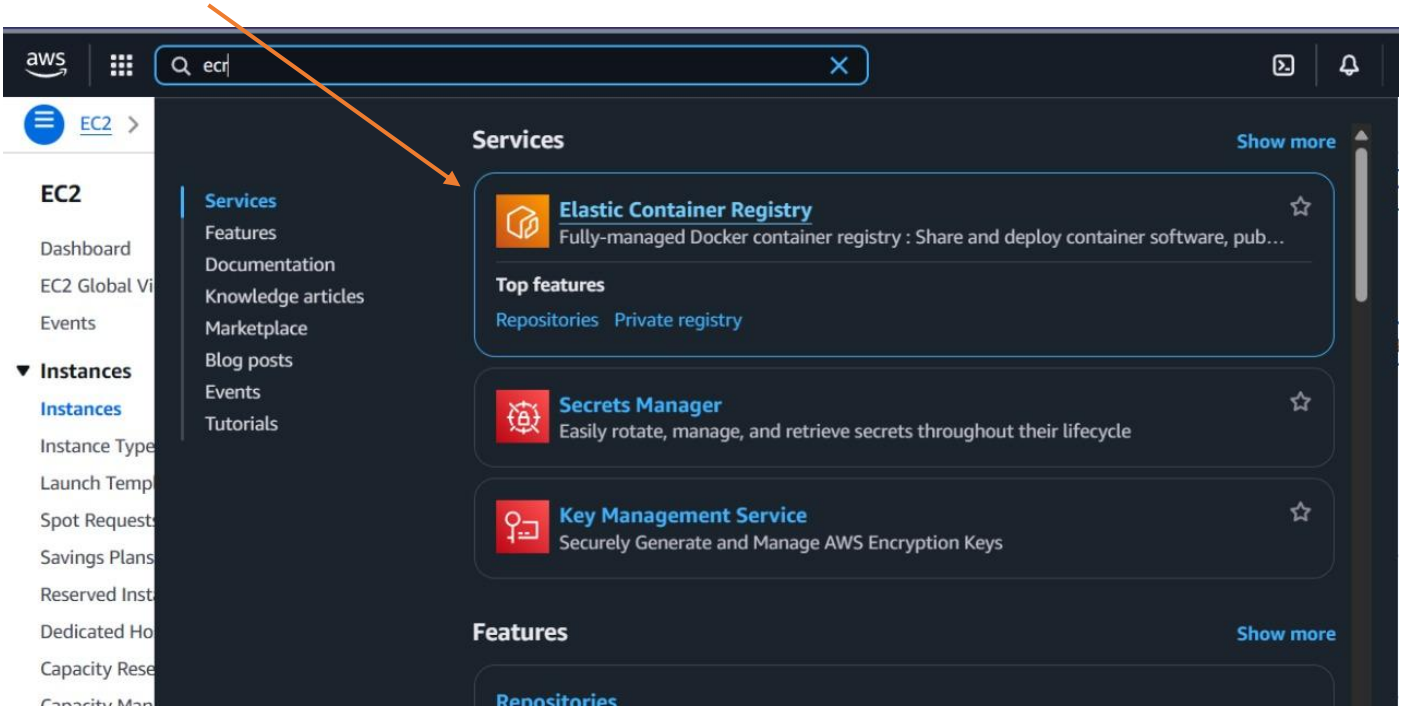
Add addition permission to the same role. Click on Add permission > attach policy.



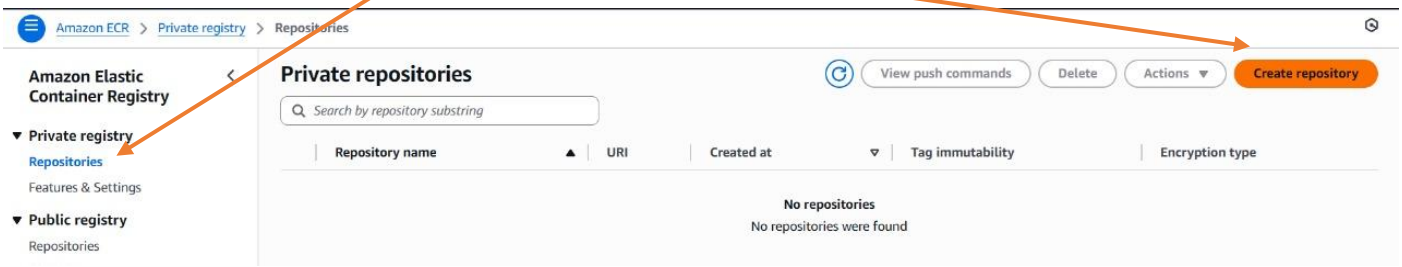
Select the below mentioned policy, click next and create.



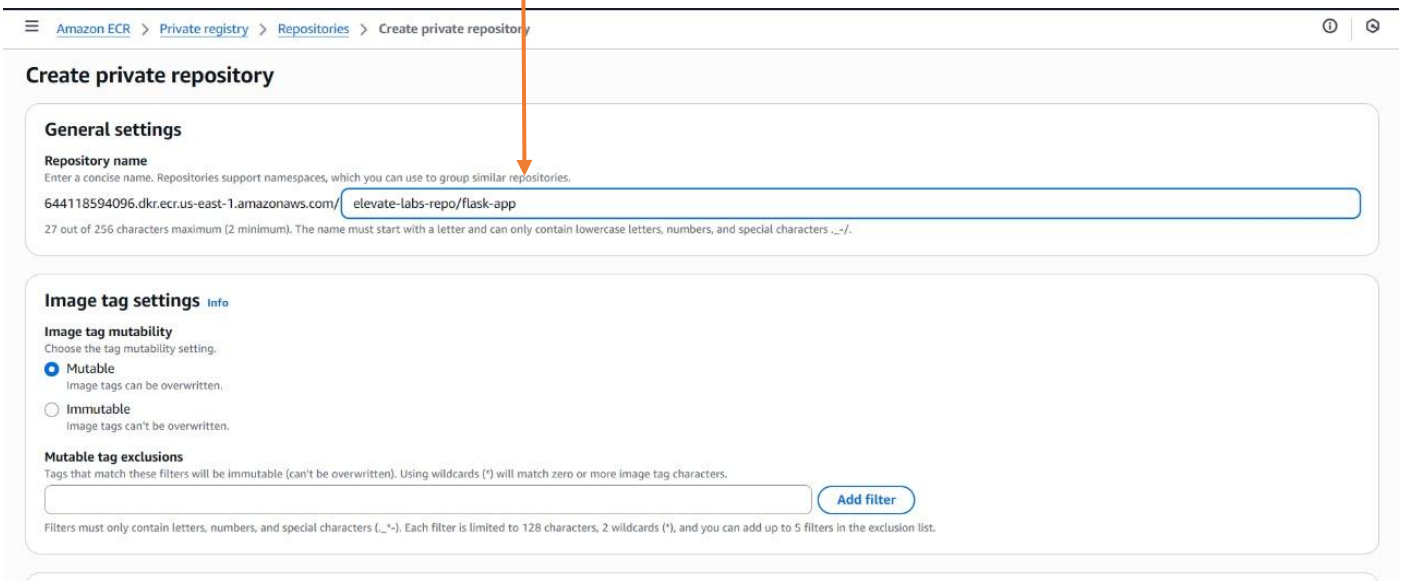
10. Go to ECR



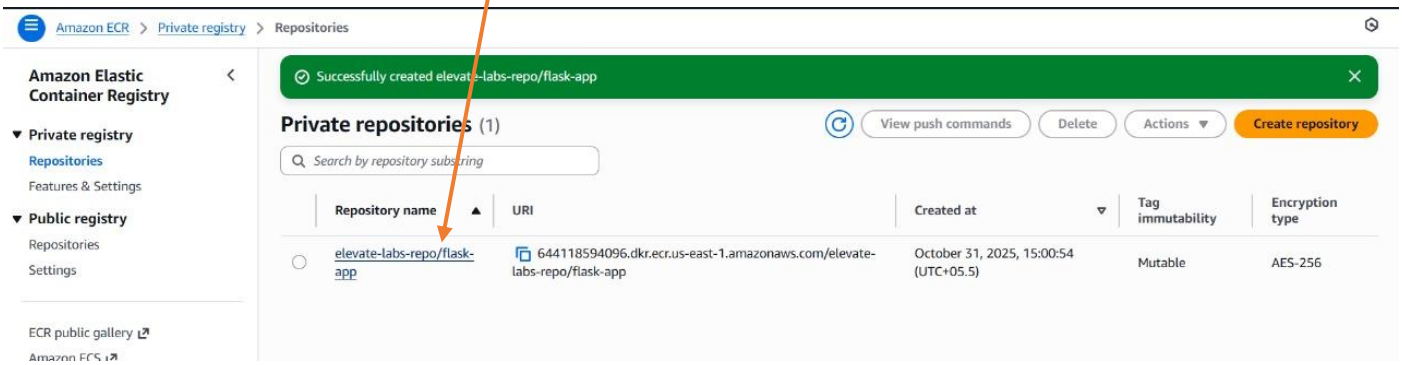
11. Click on private repositories and click “Click Repositories”.



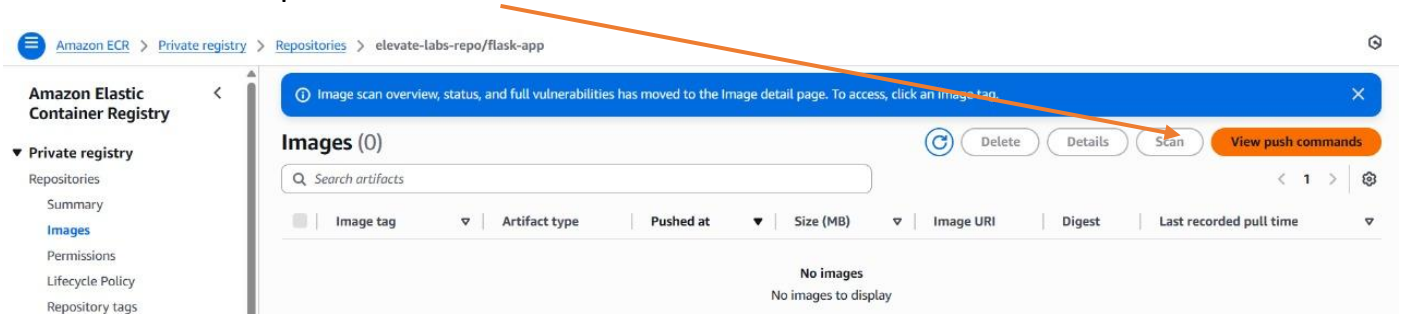
12. Name the repo example = below screenshot, leave everything default, scroll down, and click create.



13. Once created click on that repo

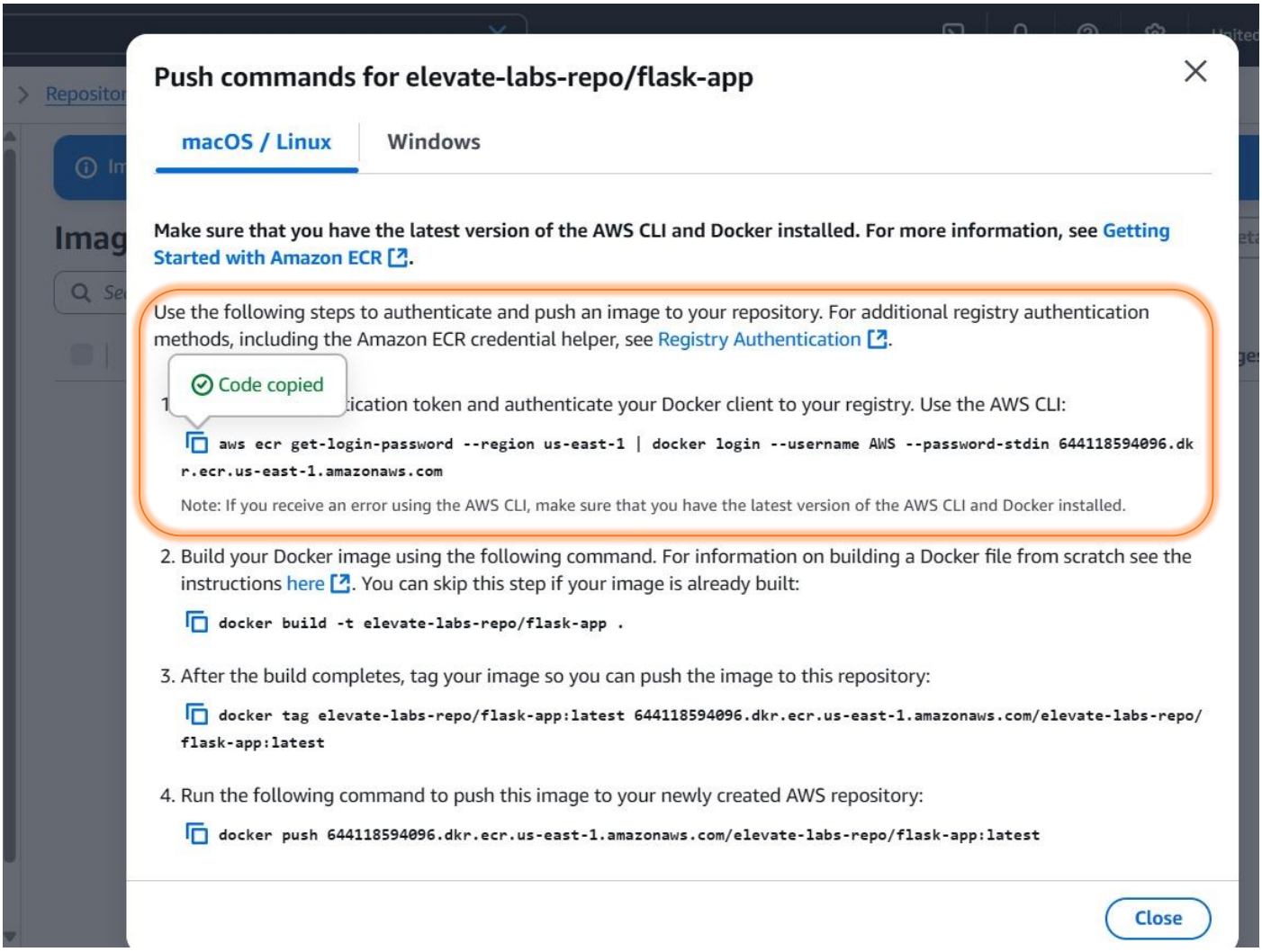


14. Click on “view push command”



15. Copy paste give command one by one. Eg

First command.



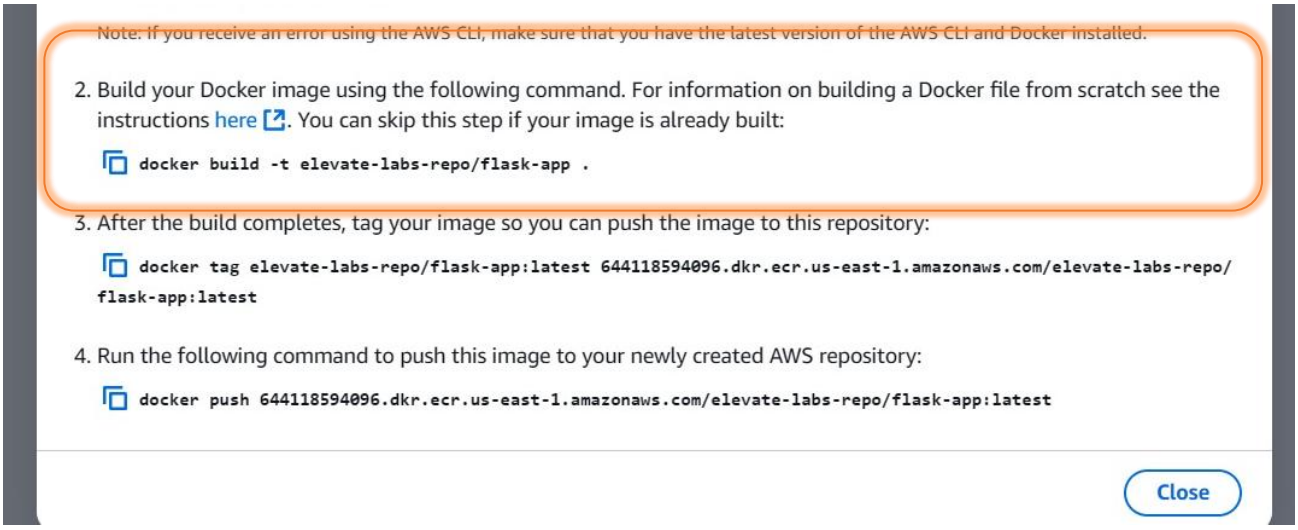
First command in terminal

```
[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]#
```

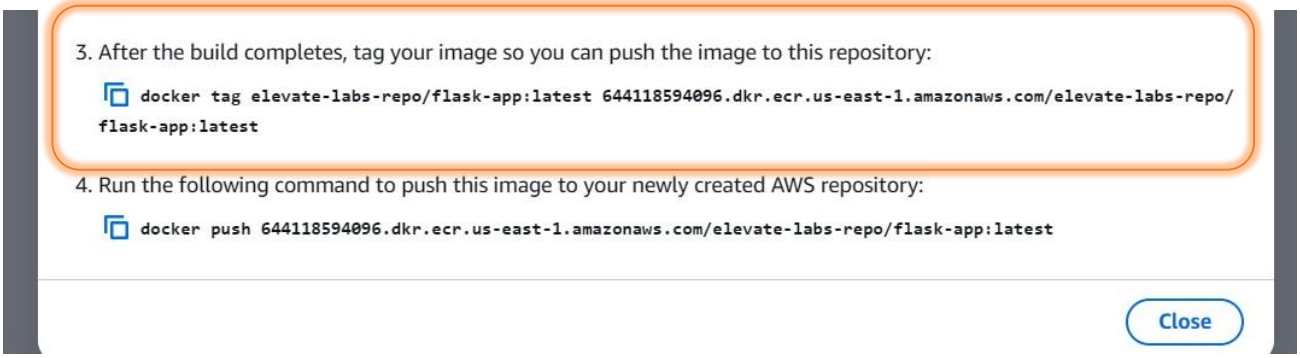
Second command



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                                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                                          0.0s
 => => transferring dockerfile: 434B                                                                                          0.0s
 => [internal] load metadata for docker.io/library/python:3.12-slim                                                           0.2s
 => [internal] load .dockerignore                                                                                             0.0s
 => => transferring context: 2B                                                                                               0.0s
 => [1/6] FROM docker.io/library/python:3.12-slim@sha256:e97cf9a2e84d604941d9902f00616db7466ff302af4b1c3c67fb7c522efa8ed9     0.0s
 => [internal] load build context                                                                                             0.0s
 => => transferring context: 356B                                                                                             0.0s
 => CACHED [2/6] WORKDIR /app                                                                                                 0.0s
 => CACHED [3/6] COPY requirements.txt requirements.txt                                                                       0.0s
 => CACHED [4/6] RUN pip install --no-cache-dir -r requirements.txt                                                           0.0s
 => CACHED [5/6] COPY app.py .                                                                                                0.0s
 => CACHED [6/6] COPY templates/ ./templates/                                                                                 0.0s
 => exporting to image                                                                                                        0.0s
 => => exporting layers                                                                                                       0.0s
 => => writing image sha256:4e98ebc3ea8a016eeb9d5993f6aa5a52978379d987ca50e7ddef4d49936d67e0                                  0.0s
 => => naming to docker.io/elevate-labs-repo/flask-app                                                                        0.0s
[root@ip-172-31-28-149 flask-app-src]#
```

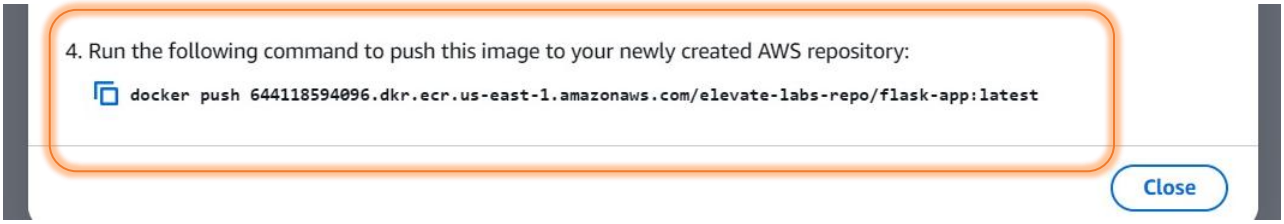
Third Command



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]#
```

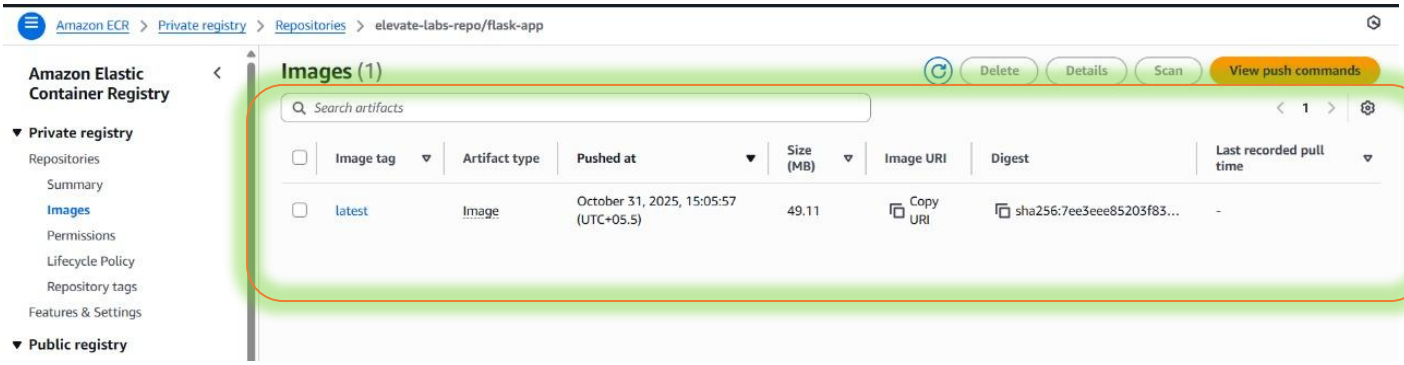
Fourth command



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
f08e023f0b9c: Pushed
9d8dfb9954dd: Pushed
7770fbb416af: Pushed
acbbee3380a1: Pushed
16dffb6a6636: Pushed
d7c97cb6f1fe: Pushed
latest: digest: sha256:7ee3eee85203f83eee1624054a1133094c4354656fb6947d501c26578036da73 size: 2198
[root@ip-172-31-28-149 flask-app-src]#
```

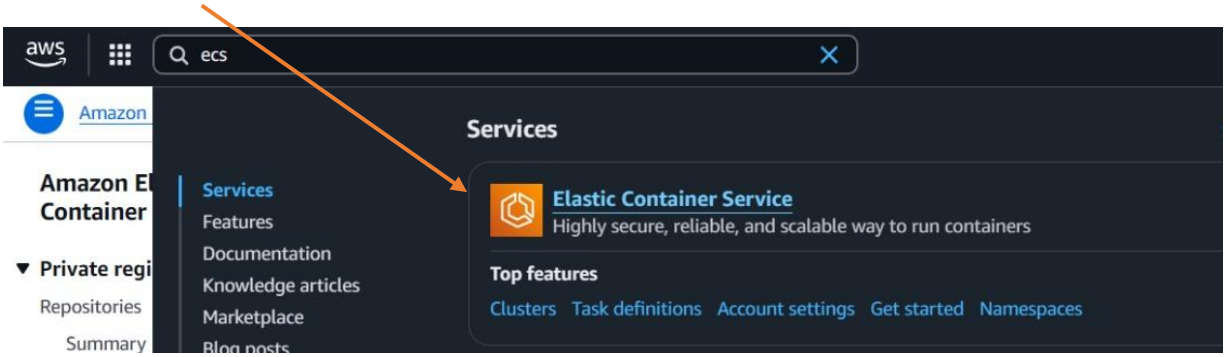
Images successfully pushed to the private repositories we created.



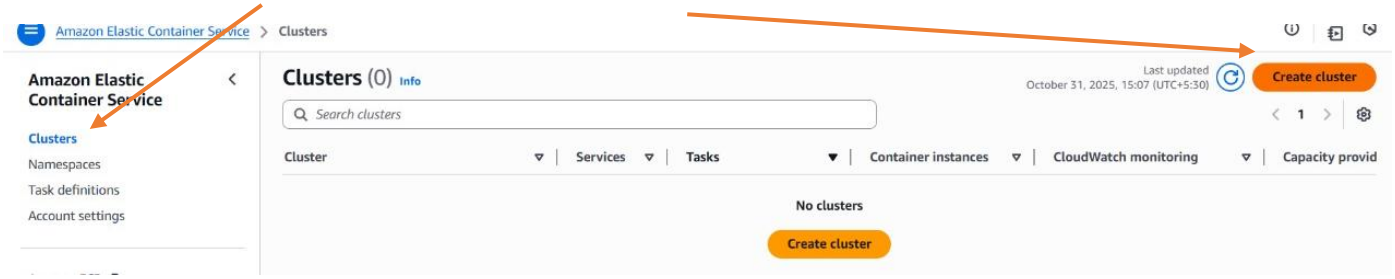
16. Create one more role step below

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

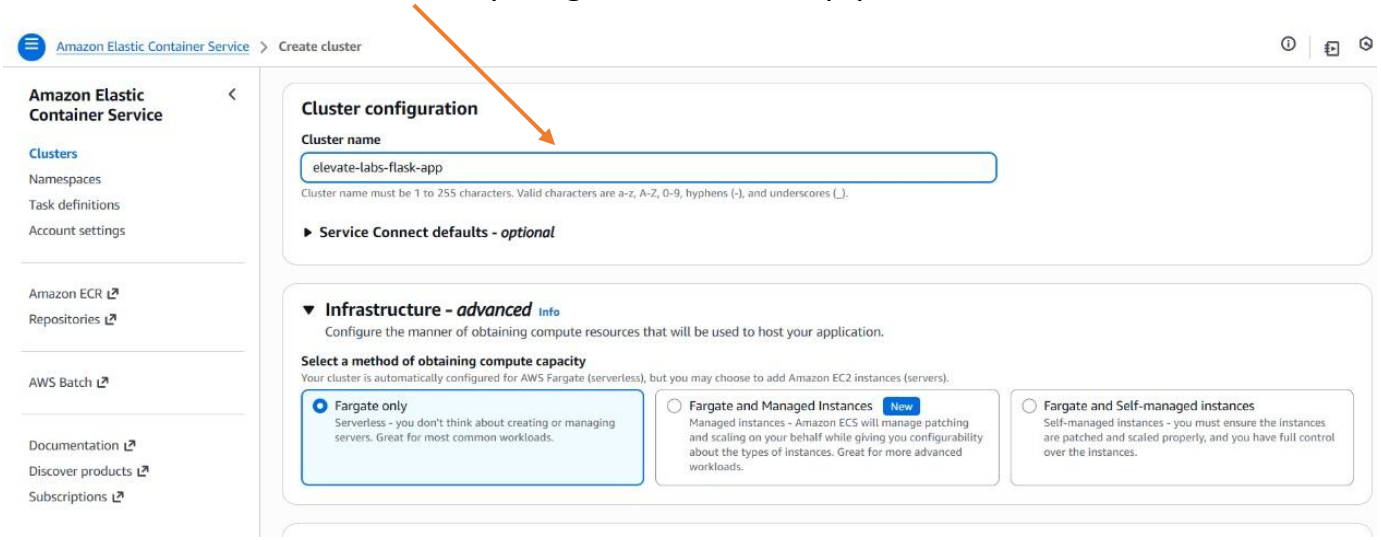
17. Go to ECS.



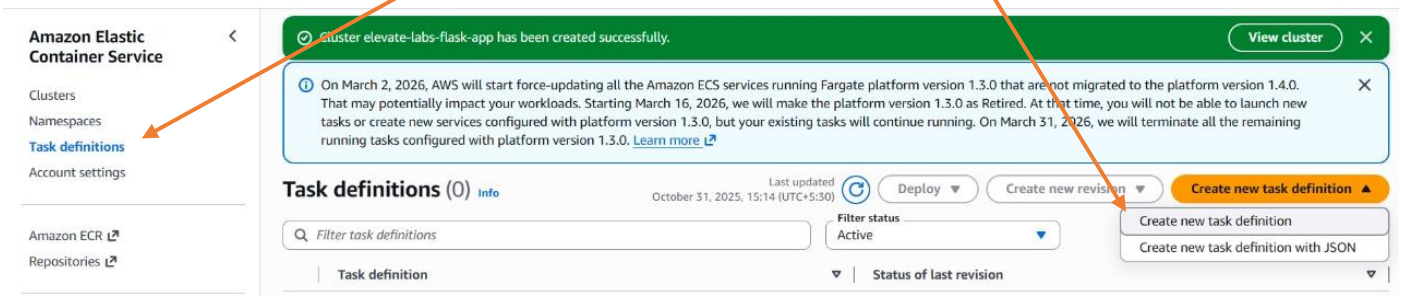
18. Select clusters in left pane and click “Create Cluster”.



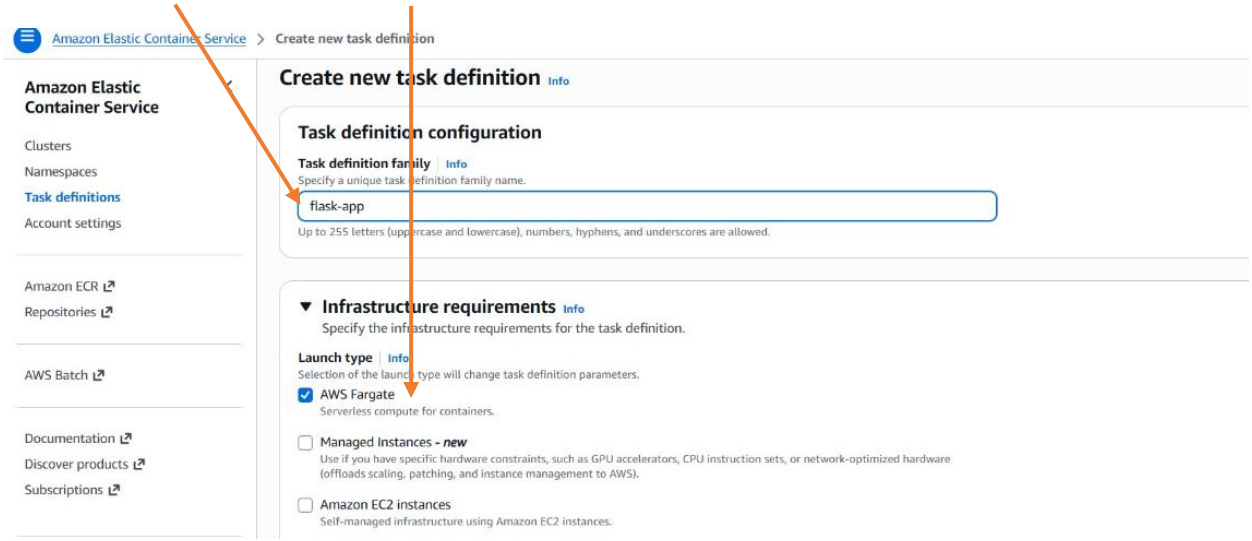
19. Name the cluster, leave everything default and simply create.



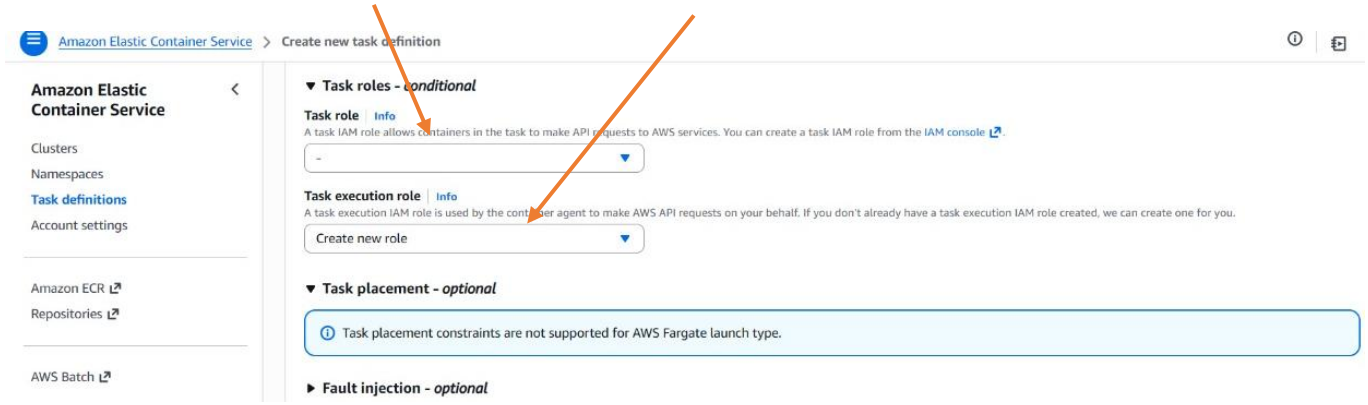
20. Go to task definition in left pane and click “Create New Task Definition”



21. Name it and select “AWS fargate”

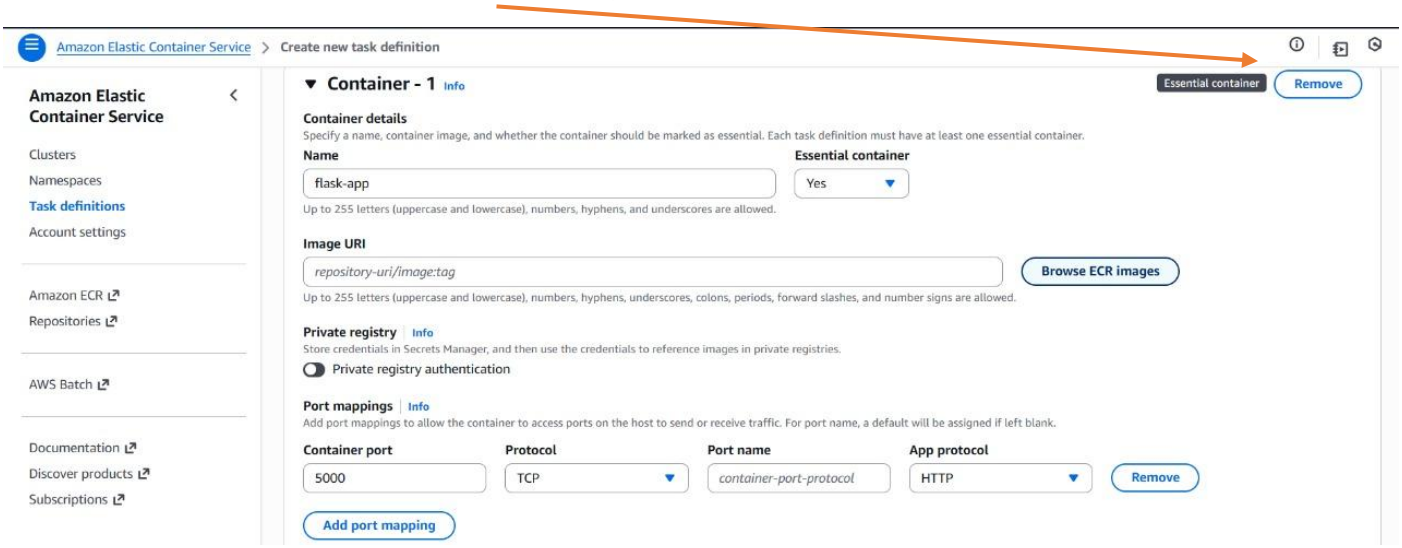


22. leave task role blank and select “Create new role” in task execution role.

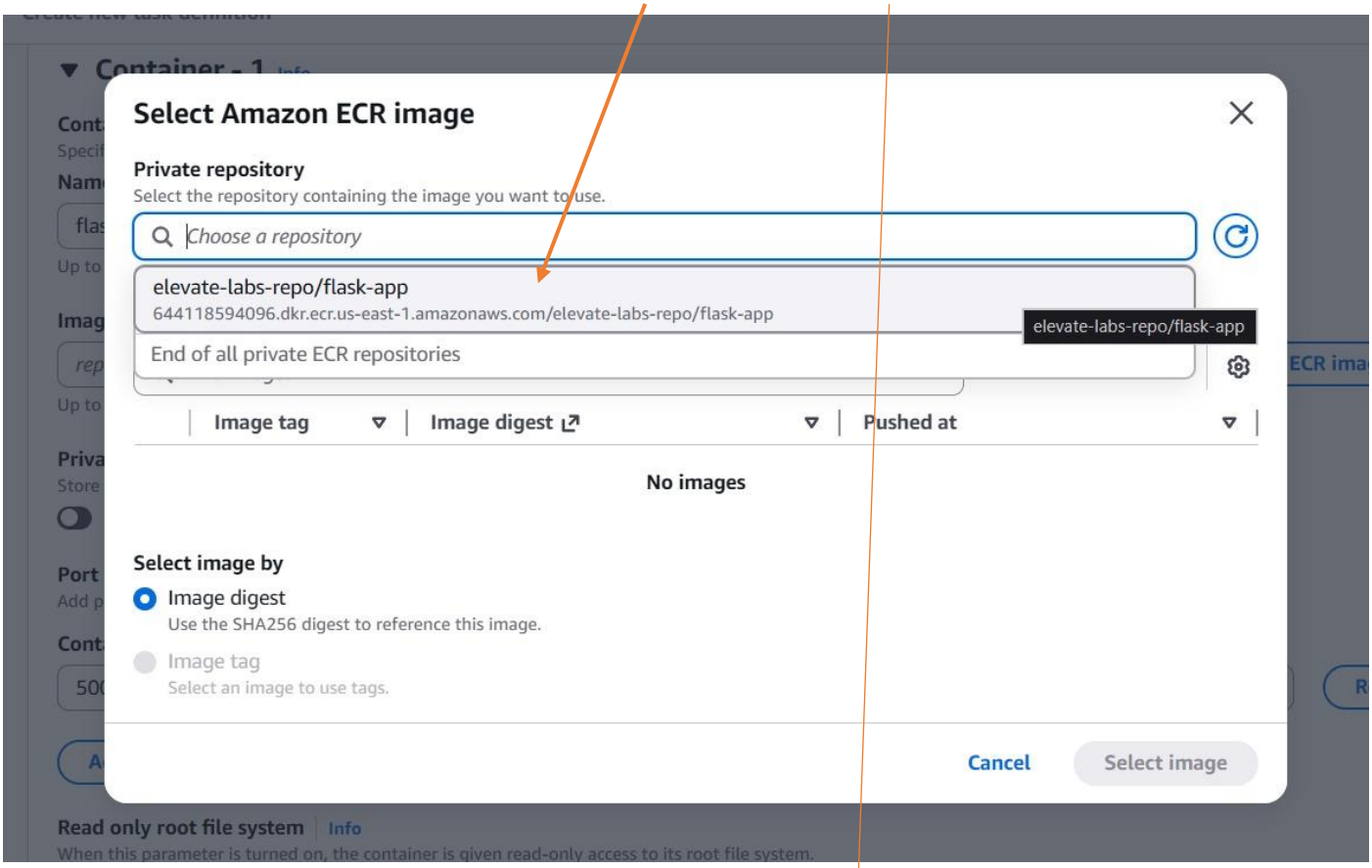


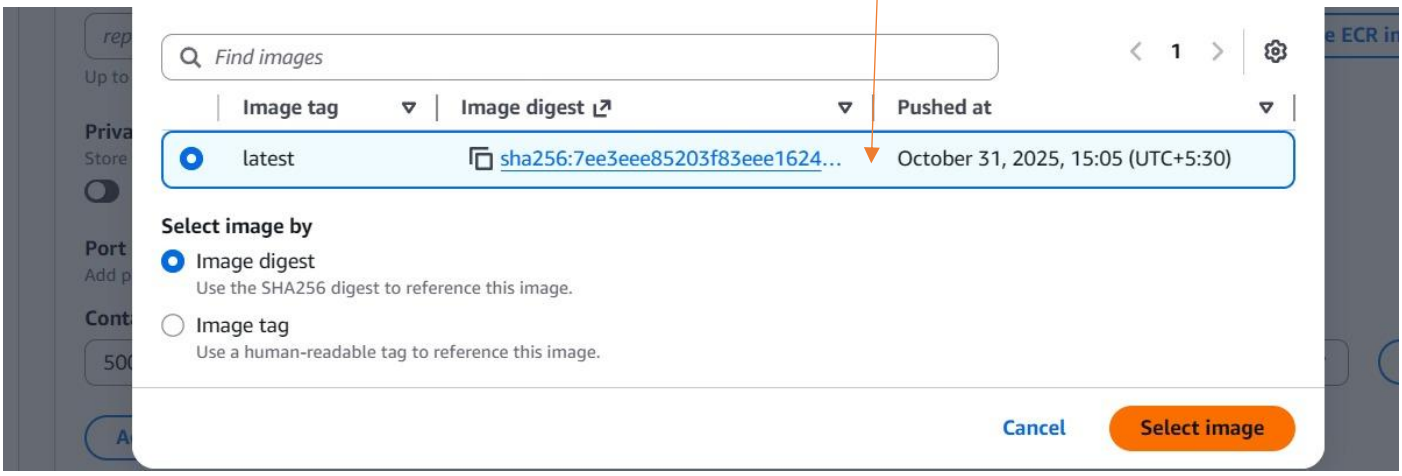
More steps below

23. In container select “Browse ECR images”

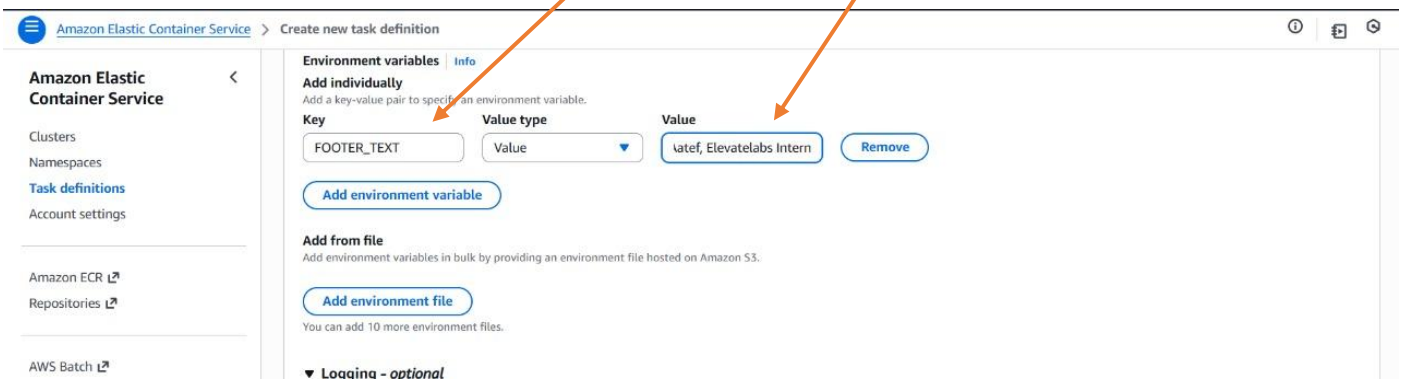


24. Select the private repo we created in step 10, and select the image

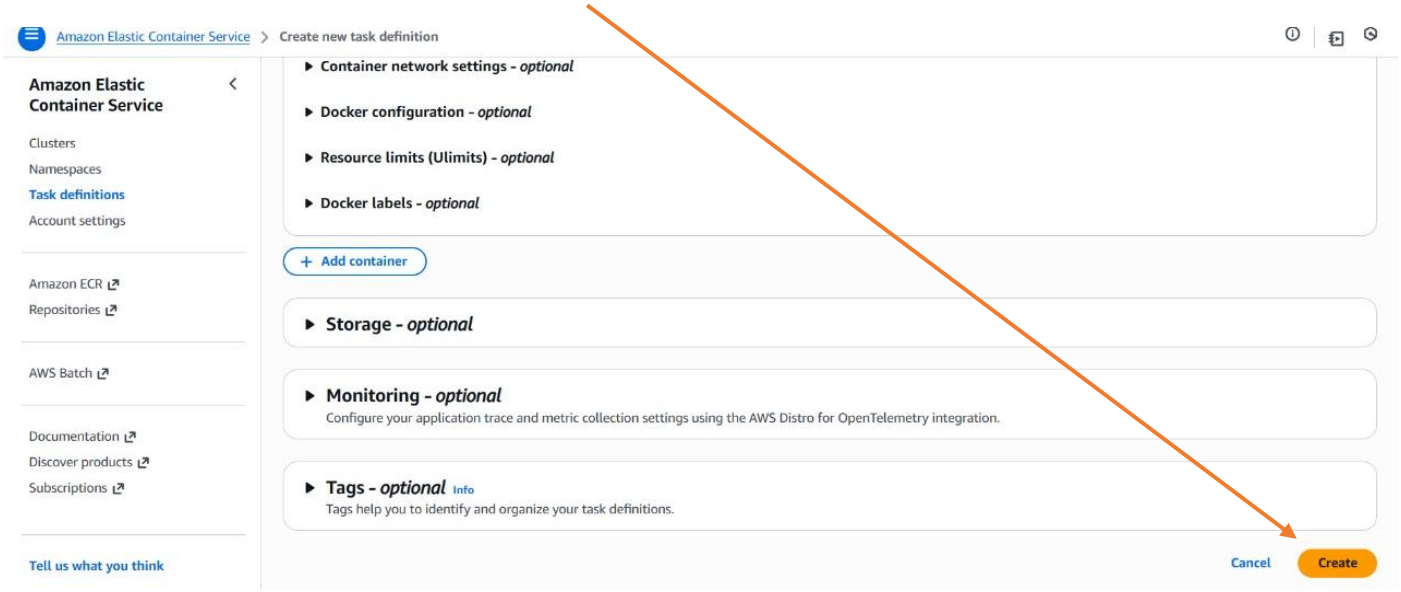




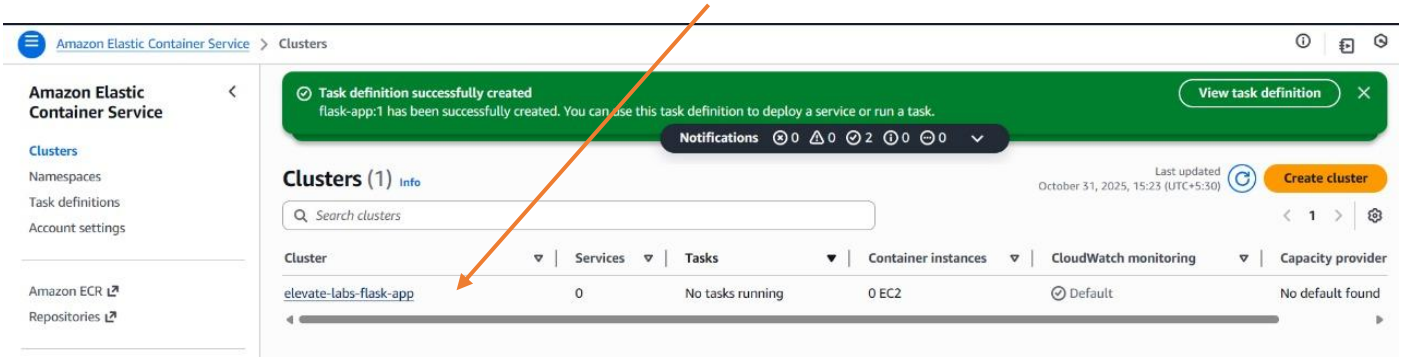
25. In environment variables, keep the key as it is and value can be anything.



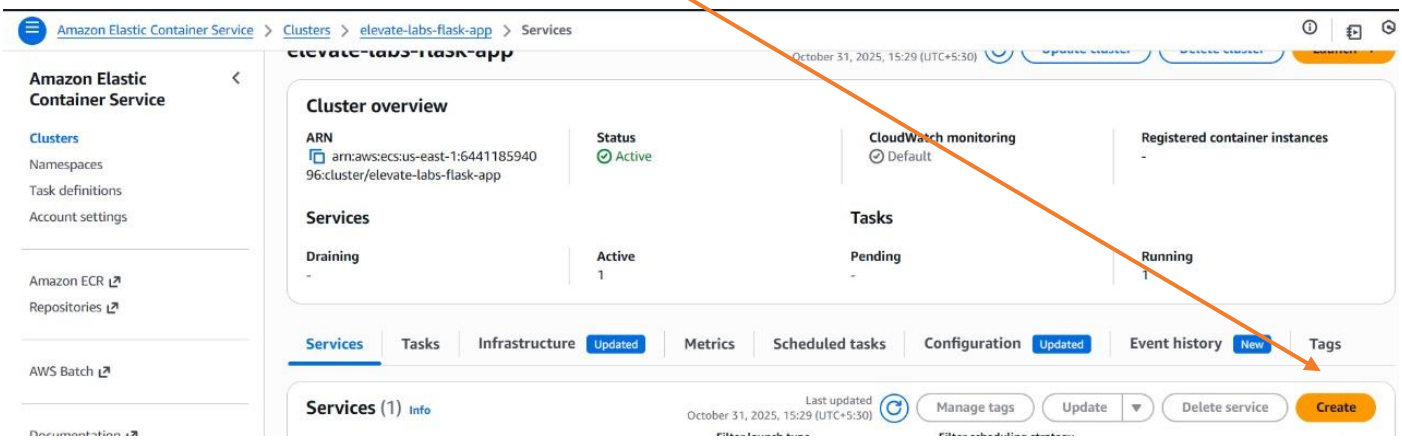
26. Leave everything default and click create.



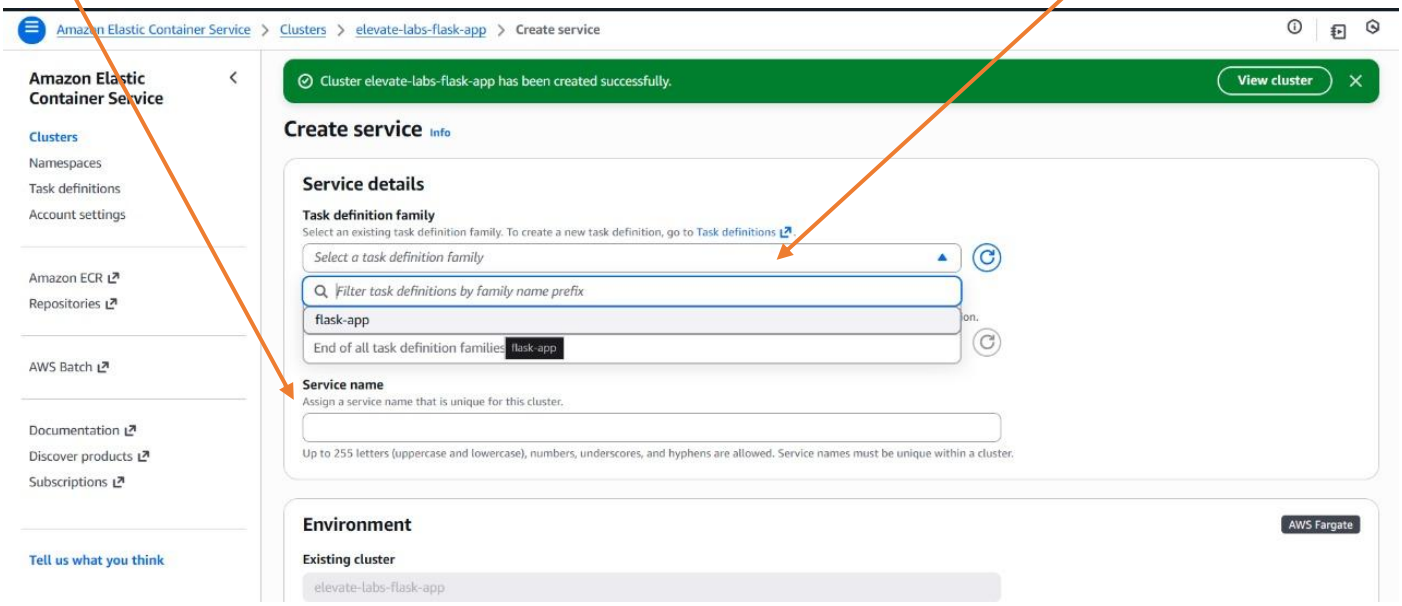
27. Navigate to clusters in left pane and click on your cluster.



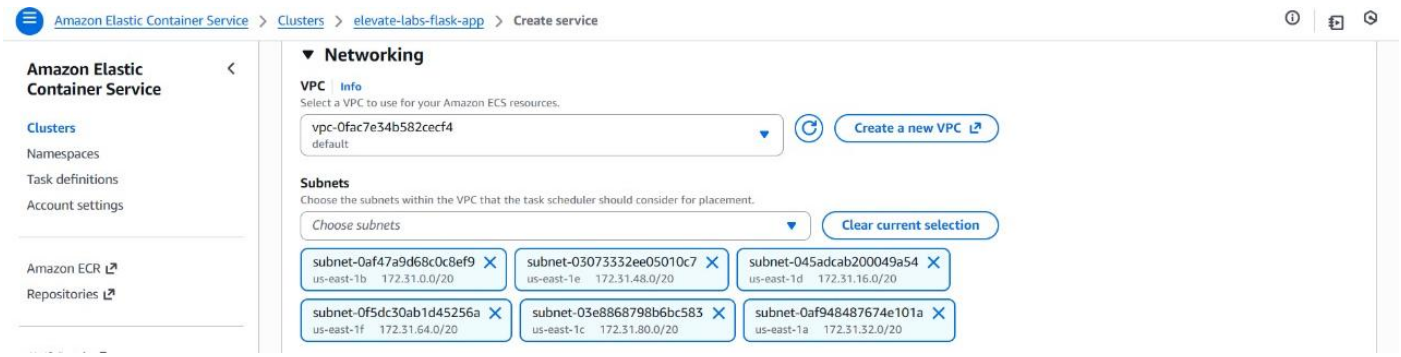
28. scroll down and in service tab click “create”.



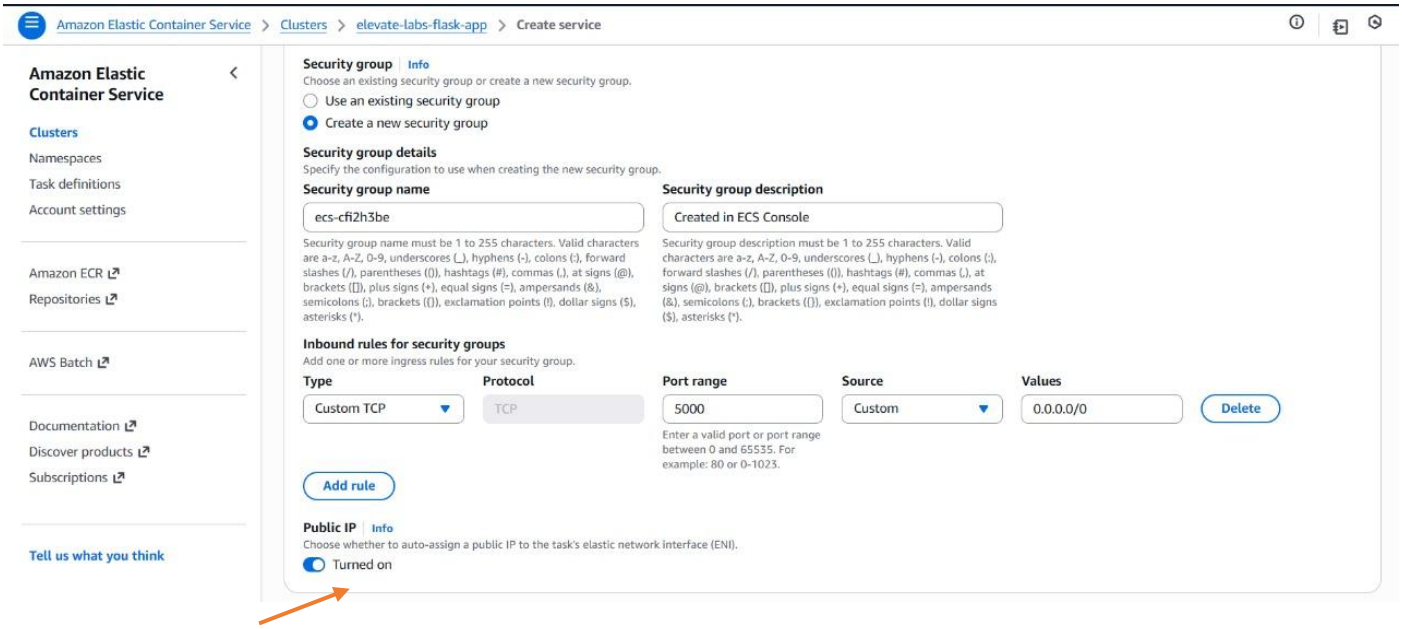
29. In task definition family select the task definition we created in step 20 and name the service.



30. In networking keep everything default.

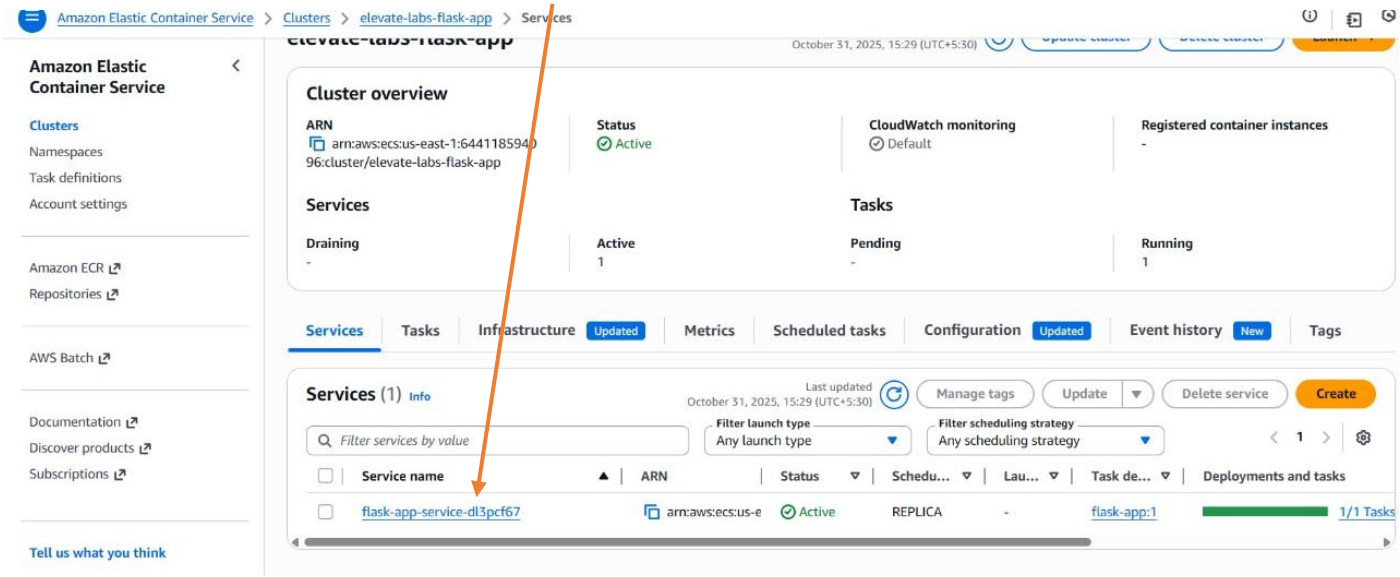


31. next create ne security group NOT default, add rule to allow traffic from everywhere for port 5000 like below, rest keep everything default and click create.

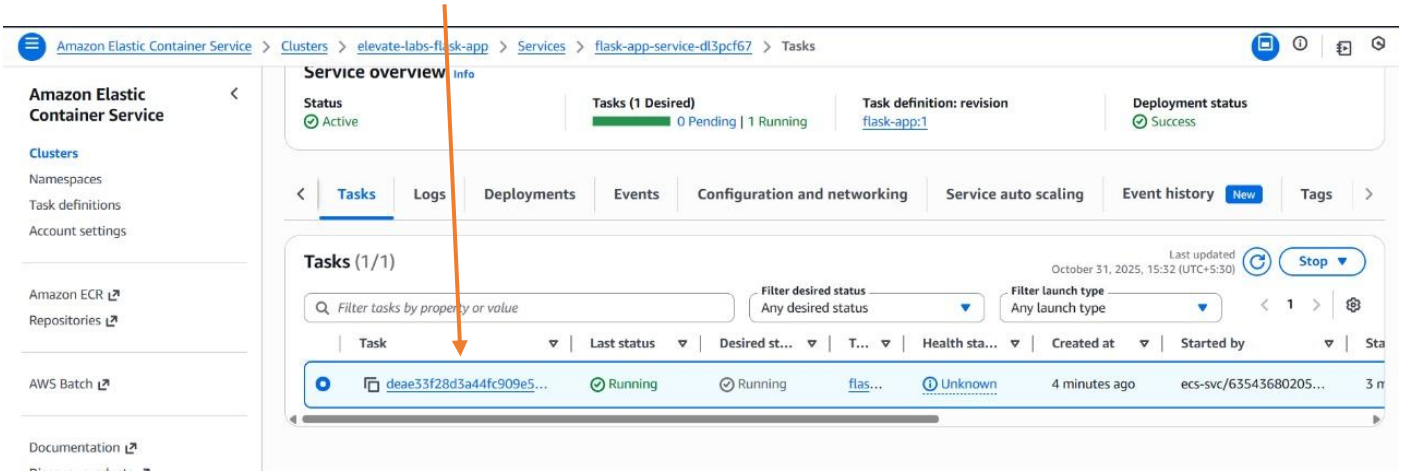


Keep public IP enabled.

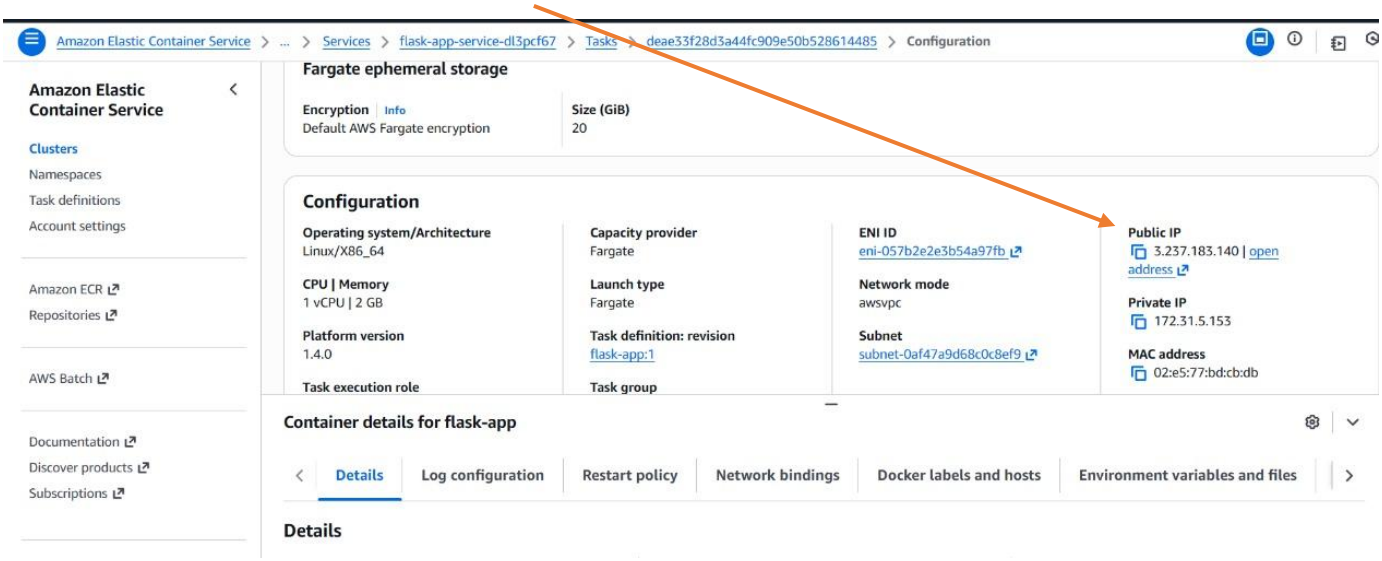
32. Once service is created click on service.



33. Then scroll down and click on tasks.



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



35. Paste the ip like this in browser.

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