

MayAkosua25 Class Project: ECS Deployment Guide

Presentation Title:

Deploying a Flask App to AWS using Docker, ECR, and ECS (Fargate)

Introduction

We are team **MayAkosua25**. In this project, we built a simple Flask web app, containerized it using Docker, pushed it to AWS Elastic Container Registry (ECR), and deployed it using ECS (Fargate). This presentation shows our process, the challenges we faced, and what we learned.

What We Did Before the Cloud

Before using AWS services, we worked on our project locally. These were the key steps:

- Built a basic **Flask web application**
- Created `index.html` inside a `templates/` folder
- Wrote backend logic using `app.py`
- Tested the app on `localhost:5000` using `flask run`
- Created a `Dockerfile` and tested the Docker container locally

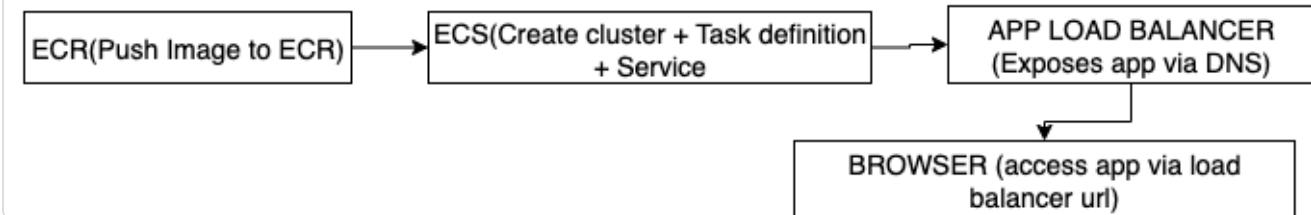
Workflow Overview

1. Build Flask app locally
2. Create a Docker image
3. Push image to Amazon ECR
4. Create ECS Cluster (Fargate)
5. Define Task
6. Deploy Service with Load Balancer
7. Access app in browser

LOCAL SECTION



AWS SECTION



✓ Step-by-Step Guide

★ Step 1: Build the Flask App

- Basic Flask app with `index.html` in a `templates/` folder
- `app.py` serves on port 5000

```

Terminal Shell Edit View Window Help
AkosuaMay25 — zsh — 179x27
.
└── bin
    └── include
        └── python3.11
            └── lib
                └── python3.11
                    └── site-packages
                        ├── MarkupSafe-3.0.2.dist-info
                        ├── distutils_hack
                        └── __pycache__
                            └── blinker
                                └── __pycache__
                                    ├── blinker-1.9.0.dist-info
                                    ├── click
                                    └── __pycache__
                                        ├── click-8.2.1.dist-info
                                        └── licenses
                            └── flask
                                └── __pycache__
                                    ├── json
                                    └── __pycache__
                                        ├── sansio
                                        └── __pycache__
                            └── flask-3.1.1.dist-info
Last login: Wed Jul 30 16:57:49 on ttys000
sadique@Abubakari-Sadiques-MacBook-Pro ~ % cd Desktop
sadique@Abubakari-Sadiques-MacBook-Pro Desktop % cd AkosuaMay25
sadique@Abubakari-Sadiques-MacBook-Pro AkosuaMay25 % tree -d
.
└── screenshots
└── templates
    └── venv
        └── bin
            └── include
                └── python3.11
                    └── lib
                        └── python3.11
                            └── site-packages
                                ├── MarkupSafe-3.0.2.dist-info
                                ├── distutils_hack
                                └── __pycache__
                                    └── blinker
                                        └── __pycache__
                                            ├── blinker-1.9.0.dist-info
                                            ├── click
                                            └── __pycache__
                                                ├── click-8.2.1.dist-info
                                                └── licenses
                            └── flask
                                └── __pycache__
                                    ├── json
                                    └── __pycache__
  
```

★ Step 2 & 3: Create Dockerfile, Build Image & Push to ECR

1. Created a Dockerfile

2. Built image:

```
docker build -t MayAkosua25-notes .
```

3. Created ECR repo: MayAkosua25-notes

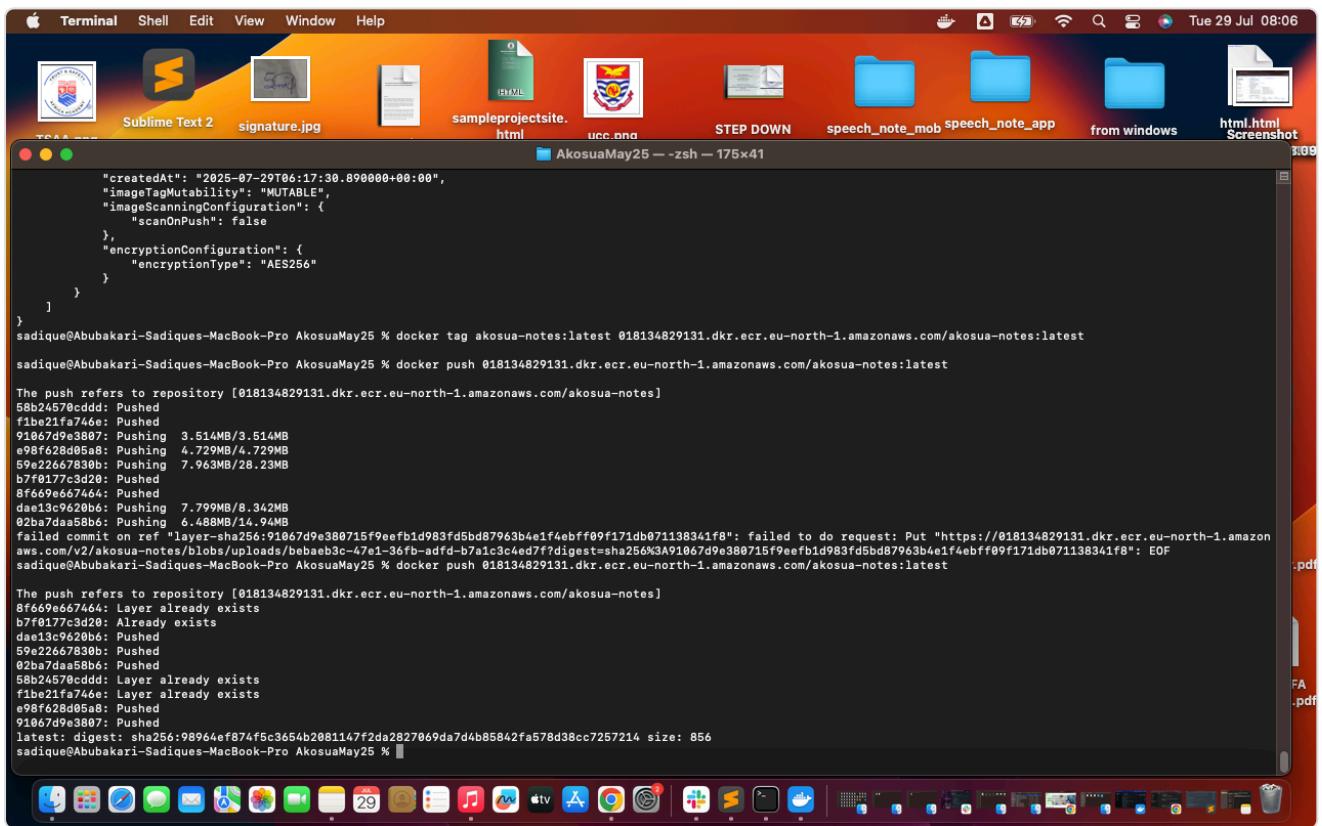
4. Logged in to ECR:

```
aws ecr get-login-password | docker login --username AWS --password-stdin .
```



5. Tagged and pushed the image:

```
docker tag MayAkosua25-notes:latest <your-ecri-url>/MayAkosua25-notes:latest
docker push <your-ecri-url>/MayAkosua25-notes:latest
```

```
Terminal Shell Edit View Window Help
Sublime Text 2 signature.jpg sampleprojectsite.html ucc.ona STEP DOWN speech_note_mob speech_note_app from windows html.html Screenshot
Tue 29 Jul 08:06
AkosuaMay25 -- zsh - 175x41
"createdAt": "2025-07-29T06:17:30.898000+00:00",
"imageTagMutability": "MUTABLE",
"imageScanningConfiguration": {
  "scanOnPush": false
},
"encryptionConfiguration": {
  "encryptionType": "AES256"
}
}
}
sadique@Abubakari-Sadiques-MacBook-Pro AkosuaMay25 % docker tag akosua-notes:latest 018134829131.dkr.ecr.eu-north-1.amazonaws.com/akosua-notes:latest
sadique@Abubakari-Sadiques-MacBook-Pro AkosuaMay25 % docker push 018134829131.dkr.ecr.eu-north-1.amazonaws.com/akosua-notes:latest
The push refers to repository [018134829131.dkr.ecr.eu-north-1.amazonaws.com/akosua-notes]
58b24570cddd: Pushed
f1be21fa746e: Pushed
91867d9e3807: Pushing 3.514MB/3.514MB
e98f628d05a8: Pushing 4.729MB/4.729MB
59e22667830b: Pushing 7.963MB/28.23MB
b7f017c73d20: Pushed
8f669e667464: Pushed
dae13c9620b6: Pushing 7.799MB/8.342MB
02ba7daa58b6: Pushing 6.488MB/14.94MB
failed commit on ref "layer-sha256:91867d9e380715f9eeffbd983fd5bd87963b4e1f4ebff09f171db071138341f8": failed to do request: Put "https://018134829131.dkr.ecr.eu-north-1.amazonaws.com/v2/akosua-notes/blobs/uploads/bebae3c-47e1-36fb-adfd-b7a1c3c4ed7f7digest=sha256%A91867d9e380715f9eeffbd983fd5bd87963b4e1f4ebff09f171db071138341f8": EOF
sadique@Abubakari-Sadiques-MacBook-Pro AkosuaMay25 % docker push 018134829131.dkr.ecr.eu-north-1.amazonaws.com/akosua-notes:latest
The push refers to repository [018134829131.dkr.ecr.eu-north-1.amazonaws.com/akosua-notes]
8f669e667464: Layer already exists
b7f017c73d20: Already exists
dae13c9620b6: Pushed
59e22667830b: Pushed
02ba7daa58b6: Pushed
58b24570cddd: Layer already exists
f1be21fa746e: Layer already exists
e98f628d05a8: Pushed
91867d9e3807: Pushed
latest: digest: sha256:98964ef874f5c3654b2081147f2da2827069da7d4b85842fa578d38cc7257214 size: 856
sadique@Abubakari-Sadiques-MacBook-Pro AkosuaMay25 %
```

The screenshot shows the Amazon ECR console with a success message: "Successfully created akosua-notes". The main area displays "Private repositories (1)" with a table. The repository details are as follows:

Repository name	URI	Created at	Tag immutability	Encryption type
akosua-notes	018134829131.dkr.ecr.eu-north-1.amazonaws.com/akosua-notes	29 July 2025, 06:17:30 (UTC-00)	Mutable	AES-256

★ Step 4: Setup the Network

- Created a second **public subnet** in a different Availability Zone
- Attached Internet Gateway and updated the route table

The screenshot shows the AWS VPC Subnets console with a success message: "You have successfully created 1 subnet: subnet-023a26bebc35da174". The main area displays "Subnets (3)" with a table. The subnet details are as follows:

Name	Subnet ID	State	VPC
-	subnet-0fa43ab2da27f14bc	Available	vpc-09bd581b9d896d644
public-subnet-1	subnet-0b3bd42a105b30701	Available	vpc-0ad656ec9c3514313 ecs...
public-subnet-2	subnet-023a26bebc35da174	Available	vpc-0ad656ec9c3514313 ecs...

★ Step 5: Create ECS Cluster

- Type: **Networking only (Fargate)**
- Cluster name: MayAkosua25-cluster

Amazon Elastic Container Service < MayAkosua25 > Services

MayAkosua25 Last updated 29 July 2025 at 12:06 (UTC)

Cluster overview

ARN arn:aws:ecs:eu-north-1:018:134829131:cluster/MayAkosua25	Status Active	CloudWatch monitoring Default	Registered container instances -
---	---	---	-------------------------------------

Services

Draining -	Tasks Pending -
Active 1	Running 1

Services | Tasks | Infrastructure | Metrics | Scheduled tasks | Configuration | Tags

Services (1) Info

Manage tags	Update	Delete service	Create
Filter services by value	Filter launch type Any launch type	Filter scheduling strategy Any scheduling strategy	< 1 >

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

★ Step 6: Create Task Definition

- Task name: MayAkosua25-task
- Runtime: Fargate
- Container:
 - Image: <your-ecr-url>/MayAkosua25-notes:latest
 - Port mapping: 5000

The screenshot shows the AWS ECS Task Definitions interface. On the left, there's a sidebar with links like Clusters, Namespaces, Task definitions (which is selected), and Account settings. The main area displays a green success message: "Task definition successfully created" followed by "MayAkosua25:1 has been successfully created. You can use this task definition to deploy a service or run a task." Below this, the task definition details are shown in a card:

- ARN:** arn:aws:ecs:eu-north-1:018134829131:task-definition/MayAkosua25:1
- Status:** ACTIVE
- Time created:** 29 July 2025 at 09:02 (UTC)
- App environment:** Fargate
- Task role:** -
- Task execution role:** ecsTaskExecutionRole
- Operating system/Architecture:** Linux/X86_64
- Network mode:** awsvpc

Below the card, there's a "Fault injection" section with a note that it's turned off. At the bottom, there are tabs for Containers (selected), JSON, Task placement, Volumes (0), Requires attributes, and Tags.

★ Step 7: Deploy Service with Load Balancer

- Deployed service inside ECS Cluster
- Service name: MayAkosua25-service
- Used an Application Load Balancer (ALB)
- Selected 2 public subnets in different AZs
- Created a Target Group on port 5000

The screenshot shows the AWS EC2 Load Balancers page. The left sidebar includes categories like Capacity Reservations, Images, AMIs, AMI Catalog, Elastic Block Store, Network & Security, Load Balancing, and Auto Scaling. Under Load Balancing, the "Load Balancers" link is selected. The main content area displays a table for load balancers:

Name	Type	Scheme	IP address type	VPC ID
MayAkosua25-lb	application	Internet-facing	IPv4	vpc-09bd581

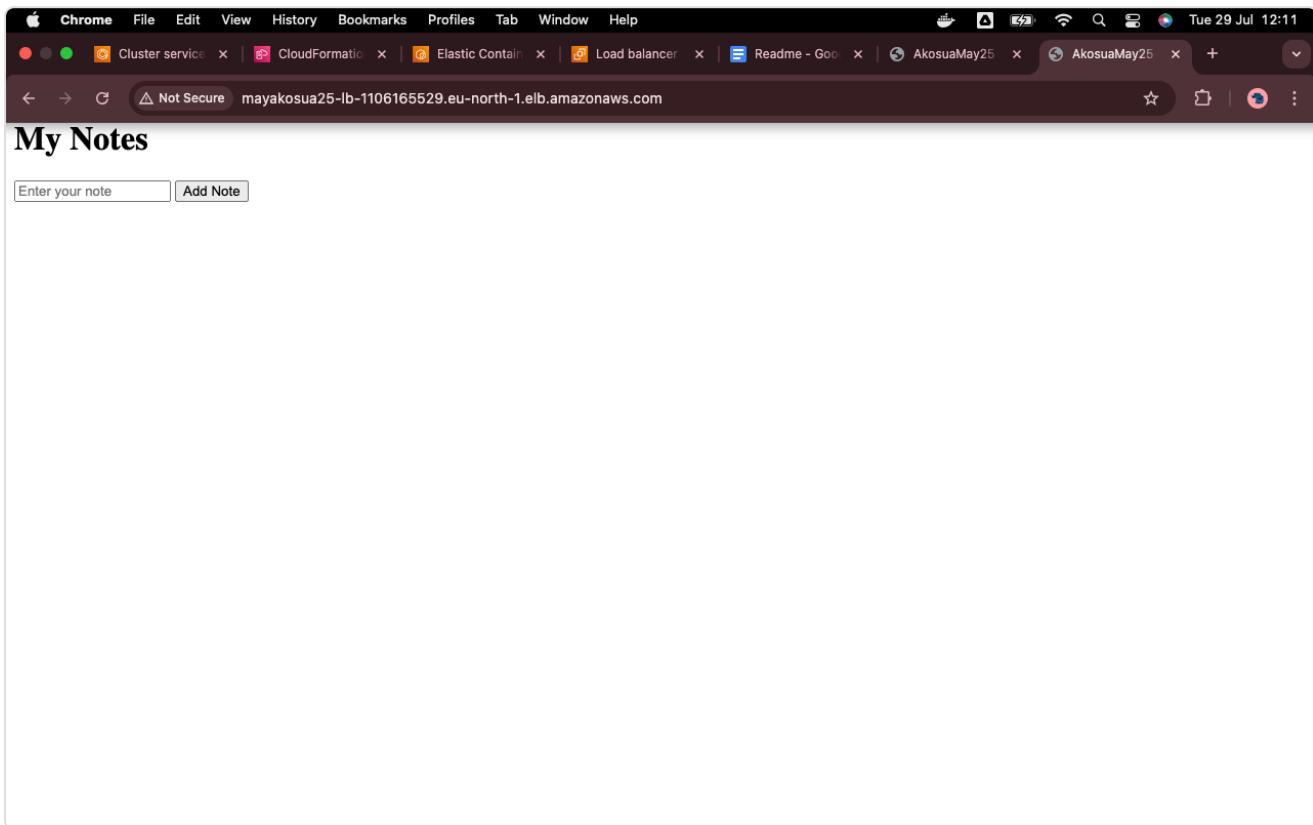
Below the table, a message says "0 load balancers selected" and "Select a load balancer above."

★ Step 8: Access the App

Load Balancer DNS:

<http://mayakosua25-lb-11061####.eu-north-1.elb.amazonaws.com>

Open this address in your browser to see the running Flask app!



✖ Errors & Fixes

- **No such file or directory** – Fixed by checking that all required files (like `requirements.txt`) existed and Docker paths were correct
- **Push access denied** – Fixed by logging in to ECR using the correct `aws ecr get-login-password` command
- **Load Balancer creation failed** – Solved by creating two public subnets in different Availability Zones and ensuring they had route access via Internet Gateway
- **Stack already exists** – Fixed by going to CloudFormation and deleting the existing stack, then re-running the deployment

🎯 Final Notes

- Always use 2 public subnets in separate AZs when using Load Balancers
- Remember to assign the correct IAM role to ECS tasks
- Check CloudFormation stacks if the UI throws errors—it controls infrastructure

What We Learned

- AWS networking and IAM can be tricky but manageable with practice
- Containers simplify app deployment across environments
- Debugging is essential to mastering cloud deployments
- We improved our skills in both development and DevOps practices

Thank you! ✨

Presented by Team Amalitech || MayAkosua25