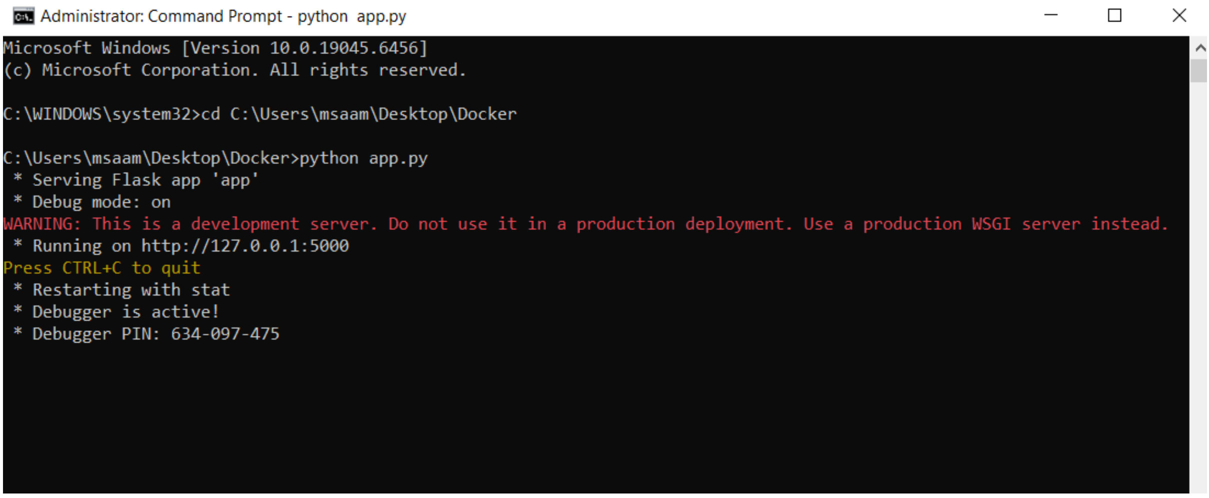
Simple Web Application Deployment Project on AWS

Please open each step by clicking on ▶ and see the explanations.

Step 1: Create a Simple Web Application

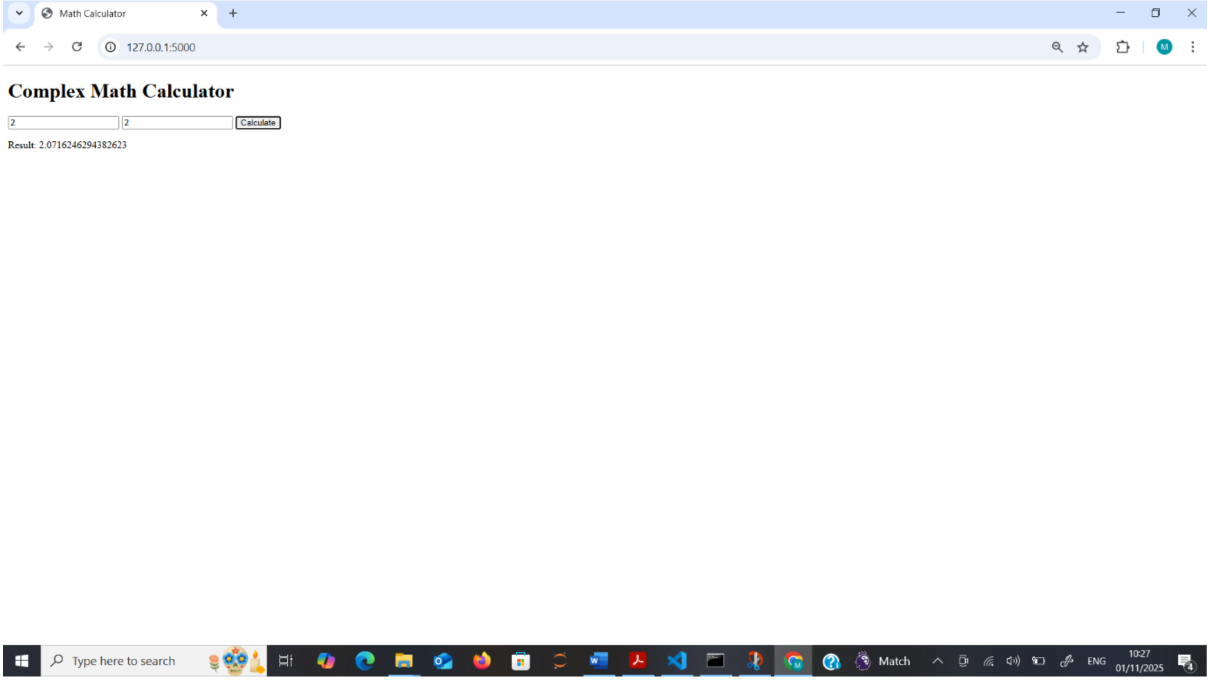
I created a basic web app using Flask. The app takes two numbers as input and calculates the formula:
$\surd(x^2 + y^2) + \sin(x * y)$
It then returns the result to the web page. The app has only two files, `app.py` (Python code) and one HTML file. No CSS or images were used to keep it simple.

The app runs locally (on my laptop) and works correctly.
In command prompt:



```
Microsoft Windows [Version 10.0.19045.6456]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\Users\msaam\Desktop\Docker

C:\Users\msaam\Desktop\Docker>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 634-097-475
```

In browser:



Complex Math Calculator

2 2

Result: 2.0716246294382623

Step 2: Containerization with Docker

I tested three methods: Docker Desktop, Play with Docker, and GitHub. I chose GitHub because it connects easily to AWS ECR. (I could also build docker image inside AWS through CodeBuild)

Note:

In `app.py`, I updated this line:

```python
app.run(debug=True)
```

to

```python
app.run(host='0.0.0.0', port=5000, debug=True)
```

This change makes the app accessible through the cloud instead of only localhost and application in deployment not be stranded because of the address.

I created a public GitHub repository named **MathApp (**Github Repo - MathApp**)**, added a `Dockerfile`, and a `requirements.txt` file listing dependencies. These files help others understand and reproduce the setup.

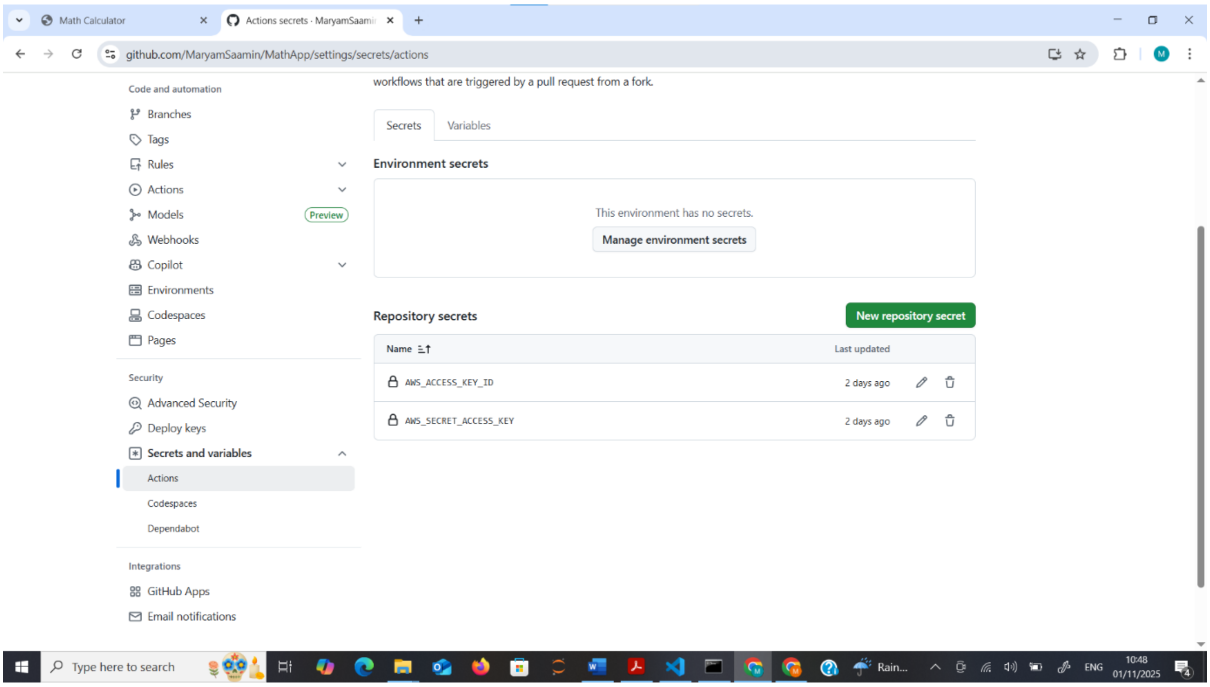
Step 3: Push Docker Image to AWS ECR

I created a new repository in AWS ECR called **mathapp-repo** (it can be also done with Iac and bash file). Then, in GitHub Actions, I added a workflow file `docker-to-ecr.yml` to build and push the Docker image to AWS ECR.

Note:

Before running the workflow, I added AWS credentials in:
`Settings > Secrets and Variables > Actions > New Repository Secret`
(`AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`), otherwise push action will be failed.

The workflow ran successfully.



github.com/MaryamSaamin/MathApp/settings/secrets/actions

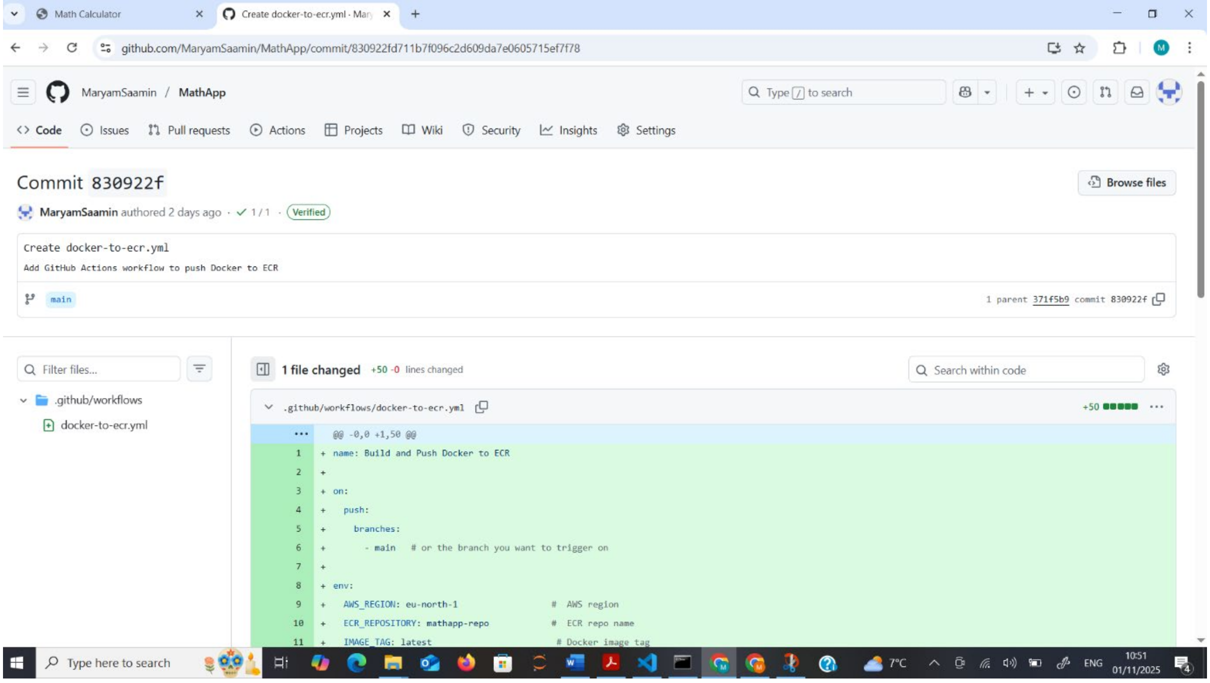
workflows that are triggered by a pull request from a fork.

Environment secrets

This environment has no secrets.

Repository secrets

Name	Last updated
AWS_ACCESS_KEY_ID	2 days ago
AWS_SECRET_ACCESS_KEY	2 days ago



github.com/MaryamSaamin/MathApp/commit/830922fd711b7f096c2d609da7e0605715ef7f78

Commit 830922f

MaryamSaamin authored 2 days ago · ✓ 1 / 1 · Verified

```
Create docker-to-ecr.yml
Add GitHub Actions workflow to push Docker to ECR
```

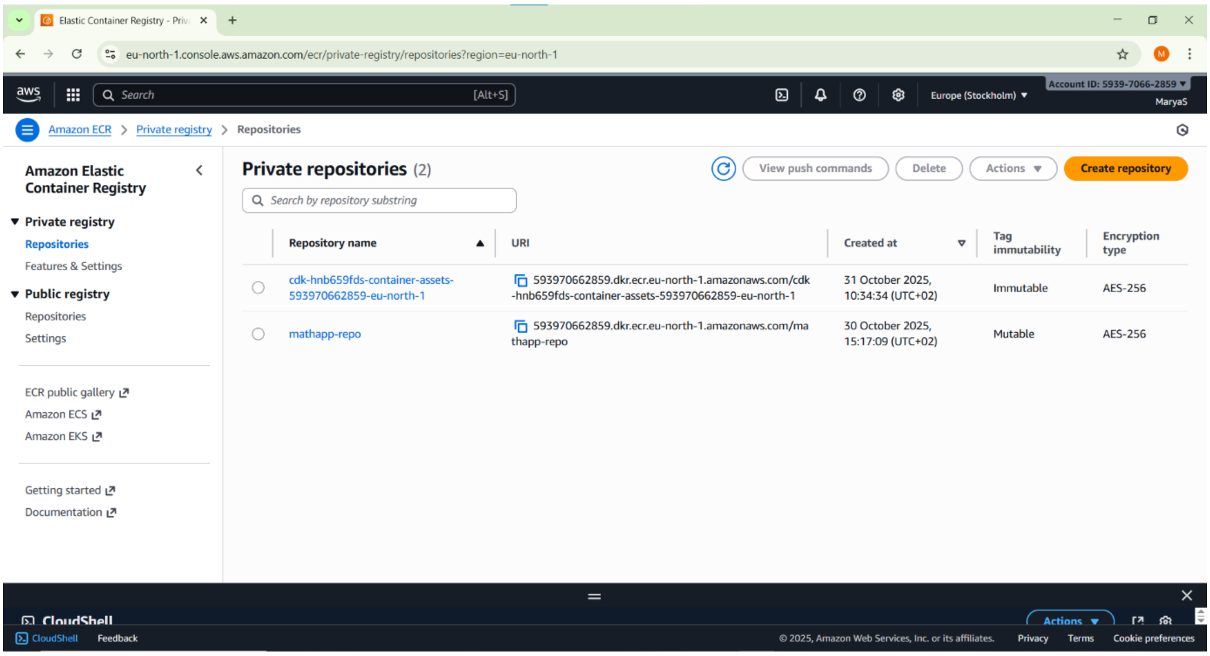
main   1 parent 371f5b9 commit 830922f

1 file changed +50 -0 lines changed

.github/workflows/docker-to-ecr.yml +50

```
@@ -0,0 +1,50 @@
+ name: Build and Push Docker to ECR
+
+ on:
+   push:
+     branches:
+       - main   # or the branch you want to trigger on
+
+ env:
+   AWS_REGION: eu-north-1                 #  AWS region
+   ECR_REPOSITORY: mathapp-repo           #  ECR repo name
+   IMAGE_TAG: latest                       # Docker image tag
```

And the Docker image was uploaded to ECR.



Elastic Container Registry - Priv
eu-north-1.console.aws.amazon.com/ecr/private-registry/repositories?region=eu-north-1
Search [Alt+S]
Europe (Stockholm)
Account ID: 5939-7066-2859
MaryaS
Amazon ECR > Private registry > Repositories
Amazon Elastic Container Registry
Private registry
Repositories
Features & Settings
Public registry
Repositories
Settings
ECR public gallery
Amazon ECS
Amazon EKS
Getting started
Documentation
View push commands
Delete
Actions
Create repository
Search by repository substring
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Private repositories (2)

	Repository name	URI	Created at	Tag immutability	Encryption type
○	cdk-hnb659fds-container-assets-593970662859-eu-north-1	593970662859.dkr.ecr.eu-north-1.amazonaws.com/cdk-hnb659fds-container-assets-593970662859-eu-north-1	31 October 2025, 10:34:34 (UTC+02)	Immutable	AES-256
○	mathapp-repo	593970662859.dkr.ecr.eu-north-1.amazonaws.com/mathapp-repo	30 October 2025, 15:17:09 (UTC+02)	Mutable	AES-256

Step 4: Deploy with AWS CDK (Infrastructure as Code)

Next, I deployed the web app on AWS using AWS CDK with Python.

Steps:

1. Install AWS CDK:

```bash
npm install -g aws-cdk
cdk --version
```

2. Create and initialize the CDK project:

```bash
mkdir flask-cdk
cd flask-cdk
cdk init app --language python
```

3. Set up Python environment:

```bash
python3 -m venv .env
source .env/bin/activate
pip install -r requirements.txt
pip install "aws-cdk-lib>=2.0.0" "constructs>=10.0.0,<11.0.0"
```

4. Edit the file `flask_cdk/flask_cdk_stack.py` with the following stack configuration (creating VPC, ECS cluster, and Fargate service with a load balancer).

Open my stack file

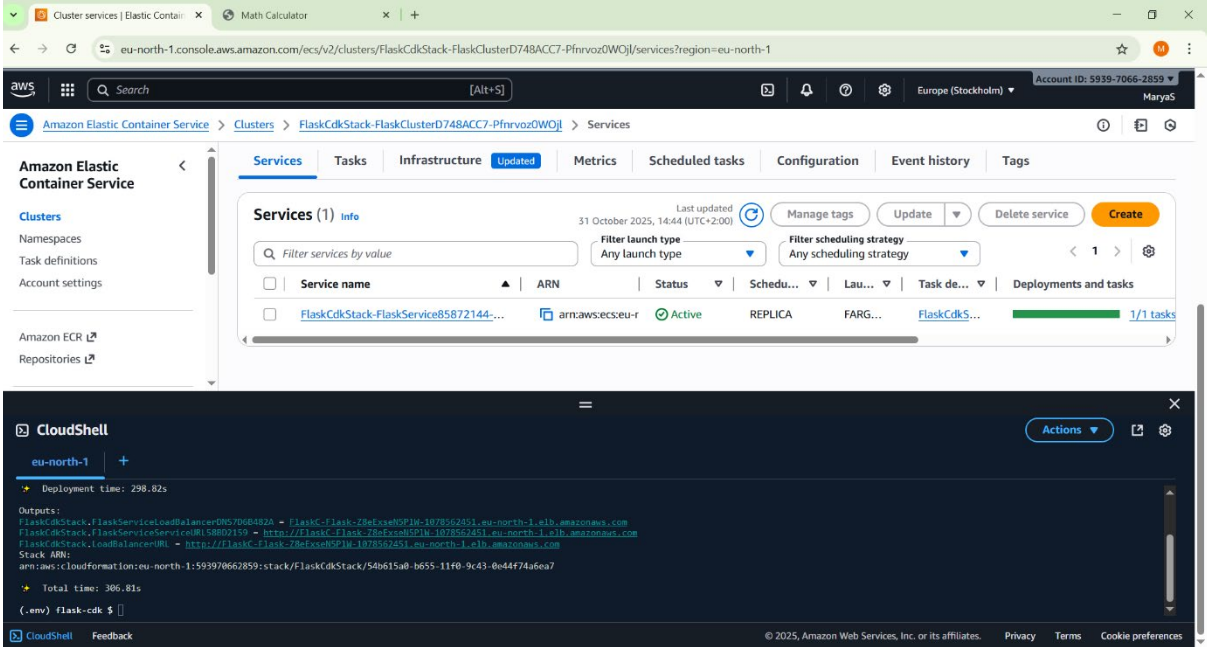
```
nano flask_cdk/flask_cdk_stack.py
```

Replace everything with this:

```
from aws_cdk import (
    Stack,
    aws_ec2 as ec2,
    aws_ecs as ecs,
    aws_ecs_patterns as ecs_patterns,
    aws_iam as iam,
    CfnOutput,
)
from constructs import Construct
class FlaskCdkStack(Stack):

    def __init__(self, scope: Construct, construct_id: str, **kwargs) -> None:
        super().__init__(scope, construct_id, **kwargs)

        # Create a new VPC
        vpc = ec2.Vpc(self, "FlaskVpc", max_azs=2)

        # Create ECS Cluster
        cluster = ecs.Cluster(self, "FlaskCluster", vpc=vpc)

        # Create ECS Task Execution Role with proper permissions
        execution_role = iam.Role(
            self, "FlaskTaskExecutionRole",
            assumed_by=iam.ServicePrincipal("ecs-tasks.amazonaws.com")
        )
        execution_role.add_managed_policy(
            iam.ManagedPolicy.from_aws_managed_policy_name(
                "service-role/AmazonECSTaskExecutionRolePolicy"
            )
        )

        # Create Fargate service behind a Load Balancer
        flask_service = ecs_patterns.ApplicationLoadBalancedFargateService(
            self,
            "FlaskService",
            cluster=cluster,
           cpu=256,
            memory_limit_mib=512,
            desired_count=1,
            public_load_balancer=True,
            task_image_options=ecs_patterns.ApplicationLoadBalancedTaskImageOptions(
                image=ecs.ContainerImage.from_registry(
                    "593970662859.dkr.ecr.eu-north-1.amazonaws.com/mathapp-repo:latest"
                ),
                container_port=5000,
                execution_role=execution_role  # <-- assign role here
            ),
        )

        # Output Load Balancer DNS
        CfnOutput(self, "LoadBalancerURL",
            value=f"http://{flask_service.load_balancer.load_balancer_dns_name}"
        )
```

5. After editing, I deployed it with:

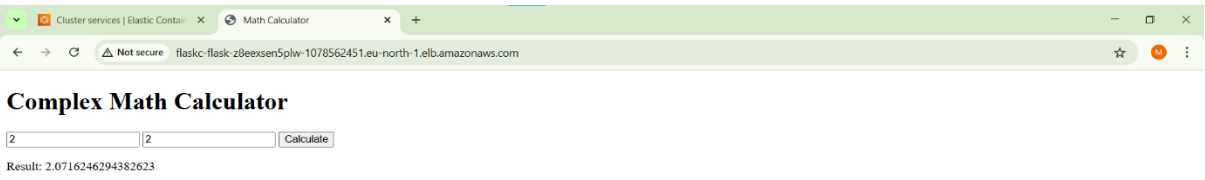
```bash
cdk bootstrap
cdk deploy
```

```

Note:

When the first deployment got stuck, I checked ECS Service events and found a missing IAM permission. After fixing the role permissions in the stack file, redeploying worked successfully (I went to ECS Service events, ECS → Clusters → Services → FlaskService → Events and I saw one task stopped and checked the **failing to pull the Docker image from ECR** because the **task execution role is missing permissions and I added to stack file**.)

The web app was live at:

AWS Console → CloudFormation → Output

http://flaskc-flask-z8eexsen5plw-1078562451.eu-north-1.elb.amazonaws.com/


Cluster services | Elastic Contai
Math Calculator
eu-north-1.console.aws.amazon.com/ecs/v2/clusters/FlaskCdkStack-FlaskClusterD748ACC7-Pfnrvoz0WOjl/services?region=eu-north-1
Search [Alt+S]
Europe (Stockholm)
Account ID: 5939-7066-2859
MaryaS
Amazon Elastic Container Service > Clusters > FlaskCdkStack-FlaskClusterD748ACC7-Pfnrvoz0WOjl > Services
Amazon Elastic Container Service
Clusters
Namespaces
Task definitions
Account settings
Amazon ECR
Repositories
Services
Tasks
Infrastructure Updated
Metrics
Scheduled tasks
Configuration
Event history
Tags
Services (1) Info
Last updated 31 October 2025, 14:44 (UTC+2:00)
Manage tags
Update
Delete service
Create
Filter services by value
Filter launch type Any launch type
Filter scheduling strategy Any scheduling strategy

| Service name | ARN | Status | Schedu... | Lau... | Task de... | Deployments and tasks |
|---|---|---|---|---|---|---|
| FlaskCdkStack-FlaskService85872144-... | arn:aws:ecs:eu-r | Active | REPLICA | FARG... | FlaskCdkS... | 1/1 tasks |

CloudShell
eu-north-1
Actions
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

```
✨ Deployment time: 298.82s

Outputs:
FlaskCdkStack.FlaskServiceLoadBalancerDNS7D6B482A = FlaskC-Flask-Z8eExseN5P1W-1078562451.eu-north-1.elb.amazonaws.com
FlaskCdkStack.FlaskServiceServiceURL58BD2159 = http://FlaskC-Flask-Z8eExseN5P1W-1078562451.eu-north-1.elb.amazonaws.com
FlaskCdkStack.LoadBalancerURL = http://FlaskC-Flask-Z8eExseN5P1W-1078562451.eu-north-1.elb.amazonaws.com
Stack ARN:
arn:aws:cloudformation:eu-north-1:593970662859:stack/FlaskCdkStack/54b615a0-b655-11f0-9c43-0e44f74a6ea7

✨ Total time: 306.81s

(.env) flask-cdk $
```
```



flaskc-flask-z8eexsen5plw-1078562451.eu-north-1.elb.amazonaws.com

Complex Math Calculator

2 | 2 | Calculate

Result: 2.0716246294382623

All the steps of deployment can be executed using the **deploy.sh** script. I initially ran them step by step, and then combined them into a single batch file.

Step 5: Clean Up Resources

To delete all AWS resources, we should run:

```bash
cdk destroy
```

For disk cleanup, I can remove caches, virtual environments, and old project folders using commands like:

```bash
find ~/.cache -type f -delete
rm -rf .env venv node_modules
pip cache purge
npm cache clean --force
```

All the steps of cleaning can be executed using the **destroy.sh** script. I initially ran them step by step, and then combined them into a single batch file. I did not destroy cdk and leave it to be checked by you.

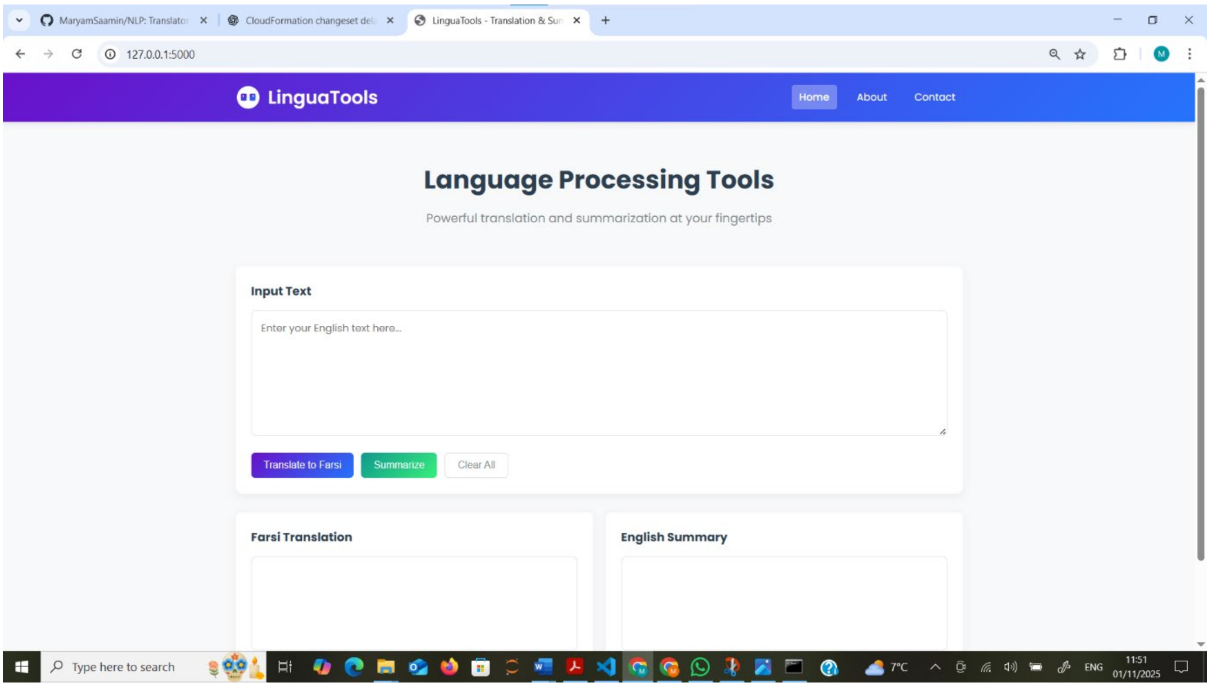
Info: Additional Hands-On

I developed more complex web applications using Machine Learning (ML) and Natural Language Processing (NLP), but because of your request, I deployed simple web application.

- ML Application: [Data Analysis GitHub Repo] (https://github.com/MaryamSaamin/Data-analysis.git)

Flask Application -Analytical Service Development and I explained everything in BDA Deployment Project Report-Saamin.pdf, and I run this flask application in **Azure Web App**.

- NLP Application: [NLP GitHub Repo] (https://github.com/MaryamSaamin/NLP.git)



MaryamSaamin/NLP: Translator
CloudFormation changeset del
LinguaTools - Translation & Sum
127.0.0.1:5000
LinguaTools
Home
About
Contact

Language Processing Tools

Powerful translation and summarization at your fingertips