Flask App Deployment on AWS with Terraform, GitHub Actions, and Custom Domain

This project demonstrates how to deploy a Flask web application using Docker, ECS (Fargate), and Terraform. It also integrates GitHub Actions for CI/CD and maps a custom domain (`23surajrc.com`) using Route 53, ACM (SSL), and Namecheap.

🚀 Project Structure

```
.
├── app/                          # Flask app folder
│   ├── Dockerfile
│   └── app.py
├── terraform/                   # Terraform infrastructure setup
│   ├── main.tf
│   ├── vpc.tf
│   ├── ecs.tf
│   ├── alb.tf
│   ├── iam.tf
│   ├── variables.tf
│   ├── outputs.tf
│   └── backend-setup.tf
├── .github/workflows/
│   └── deploy.yml               # GitHub Actions CI/CD pipeline
├── README.md
└── .gitignore
```

PROF

🐍 Flask App (app/app.py)

```python
from flask import Flask
app = Flask(__name__)

@app.route('/')
def home():
    return "Hello from Flask on ECS!"

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

🐳 Dockerfile

```
FROM python:3.9-slim
WORKDIR /app
COPY . .
RUN pip install flask
CMD ["python", "app.py"]
```

👷 Terraform Infrastructure

1. Initialize Backend (S3 + DynamoDB for tfstate)

```
# backend-setup.tf
resource "aws_s3_bucket" "tfstate" {
  bucket = "tfstate-suraj2310-20250716"
  versioning { enabled = true }
  force_destroy = true
  tags = { Name = "Terraform State Bucket", Environment = "dev" }
}

resource "aws_s3_bucket_server_side_encryption_configuration" "tfstate"
{
  bucket = aws_s3_bucket.tfstate.id
  rule {
    apply_server_side_encryption_by_default {
      sse_algorithm = "AES256"
    }
  }
}

resource "aws_dynamodb_table" "tf_locks" {
  name         = "terraform-locks"
  billing_mode = "PAY_PER_REQUEST"
  hash_key     = "LockID"
  attribute {
    name = "LockID"
    type = "S"
  }
  tags = {
    Name = "Terraform Locks Table"
    Environment = "dev"
  }
}
```

PROF

Then run:

```
terraform init -backend-config="bucket=tfstate-suraj2310-20250716" \
               -backend-config="key=terraform.tfstate" \
               -backend-config="region=ap-south-1" \
               -backend-config="dynamodb_table=terraform-locks"
```

2. Provision the Infrastructure

```
terraform plan
terraform apply
```

This will:

- Create a VPC with public and private subnets
- Create an ECS Cluster with a Fargate service
- Create an ALB to expose the service

GitHub Actions CI/CD

Create `.github/workflows/deploy.yml`:

```
name: Deploy Flask App to ECS

on:
  push:
    branches:
      - main

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Configure AWS Credentials
        uses: aws-actions/configure-aws-credentials@v2
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          aws-region: ap-south-1

      - name: Build & Push Docker image to ECR
        run: |
          # Replace with your own logic for ECR login & push
          echo "Build and push steps here"
```

PROF

```
      - name: Terraform Apply
        run: |
          cd terraform
          terraform init
          terraform apply -auto-approve
```

🔗 Mapping Custom Domain (23surajrc.com) with SSL

1. Route 53 Hosted Zone

- Create a hosted zone for `23surajrc.com` in Route 53.
- Note the NS records and update them in **Namecheap** under **Domain > Advanced DNS**.

2. ACM (SSL Certificate)

- Request a certificate for:
  - `23surajrc.com`
  - `www.23surajrc.com`
- Choose **DNS validation**.
- Add the provided **CNAME** records into your Route 53 hosted zone.
- Wait until status = `Issued`.

3. ALB HTTPS Listener

- Create an HTTPS listener (port 443) on your Application Load Balancer.
- Attach the issued ACM certificate.
- Forward traffic to your ECS target group.

PROF

4. DNS Records in Route 53

Create:

- **A Record** for `23surajrc.com` → Alias → your ALB DNS name
- **A Record** for `www.23surajrc.com` → Alias → your ALB DNS name

Now both `https://23surajrc.com` and `https://www.23surajrc.com` should work securely.

✅ Summary

Feature	Done
Flask App in Docker	✅
ECS (Fargate) Deployment	✅

Feature	Done
Terraform Infra Provision	✅
GitHub Actions CI/CD	✅
Domain Mapping via Route 53	✅
SSL Certificate via ACM	✅

📝 Notes

- Make sure ports 80 and 443 are open in the ALB security group.
- ACM certificates must be in **us-east-1** for CloudFront, but in your region (e.g., ap-south-1) for ALB.
- Use `terraform destroy` to tear down the infra when done.

:handshake: Contributing

Feel free to fork and improve this project. Pull requests are welcome!