Medusa-ECS-Fargate-terraform-CI/CD

To implement Infrastructure as code for the Medusa open-source headless commerce backend using Terraform and deploy it on AWS ECS/Fargate, follow these steps:

Terraform setup:

1. Create a vpc: vpc with subnets, Internet gateway, and route tables.

2. Create ECS Cluster: Define an ECS Cluster and configure Fargate as the launch type.

3. Task Definition: Create an ECS task definition with Docker container specifications for Medusa.

4. ECS Service: Set up an ECS service that runs the task definition with load balancing and auto-scaling as needed.

5. Security Groups: Define security groups for your ECS service, allowing necessary inbound and outbound traffic.

6. IAM Roles: Define IAM roles with permissions for ECS tasks, including access to ECR etc.

GitHub Action CI/CD Pipeline:

1. Trigger: Define triggers for the pipeline, such as pushing to the main branch.

2. Checkout Code: Use actions/checkout@v2 to pull the latest code from the repository

3. Setup Terraform: Install Terraform and initialize it using terraform init.

4. Build & Push Docker Image: Build Medusa Docker image and push it to an AWS ECR repository.

5. Deploy with Terraform: Apply the Terraform configuration to deploy the infrastructure on AWS.

6. Update ECS Service: Use Terraform or AWS CLI to update the ECS service with the latest Docker image.

GitHub Action Workflow:

I add a file **deploy.yml** inside workflow

```
 name: Deploy to AWS ECS
on:
  push:
   branches:
     - main
  workflow_dispatch:

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Set up Terraform
      uses: hashicorp/setup-terraform@v1
      with:
        terraform_version: 1.0.0

    - name: Terraform Init
      run: |
        terraform init
      working-directory: ./Eks-terraform

    #- name: Terraform plan
     # run: |
     #   terraform plan

    - name: Terraform validate
      run: |
        terraform validate
      working-directory: ./Eks-terraform

    - name: Terraform Apply
      run: |
        terraform apply -auto-approve
      env:
        AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY }}
        AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
      working-directory: ./Eks-terraform

    # - name: Log in to Amazon ECR
    #   id: ecr_login
    #   uses: aws-actions/amazon-ecr-login@v1
    #   with:
    #     region: us-east-1



    - name: Log in to Amazon ECR
      run: |
        aws ecr get-login-password --region us-east-1 | \
        docker login --username AWS --password-stdin xxxxxxxxxx.dkr.ecr.us-east-
  1.amazonaws.com  # replace here ecr login cmd
      env:
        AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY }}
        AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
        AWS_REGION: us-east-1
        TERRAFORM_CLI_PATH: /home/runner/work/_temp/d747af65-d62e-4200-b1eb-
  3693991067e9

    - name: Build and Tag Docker image
```

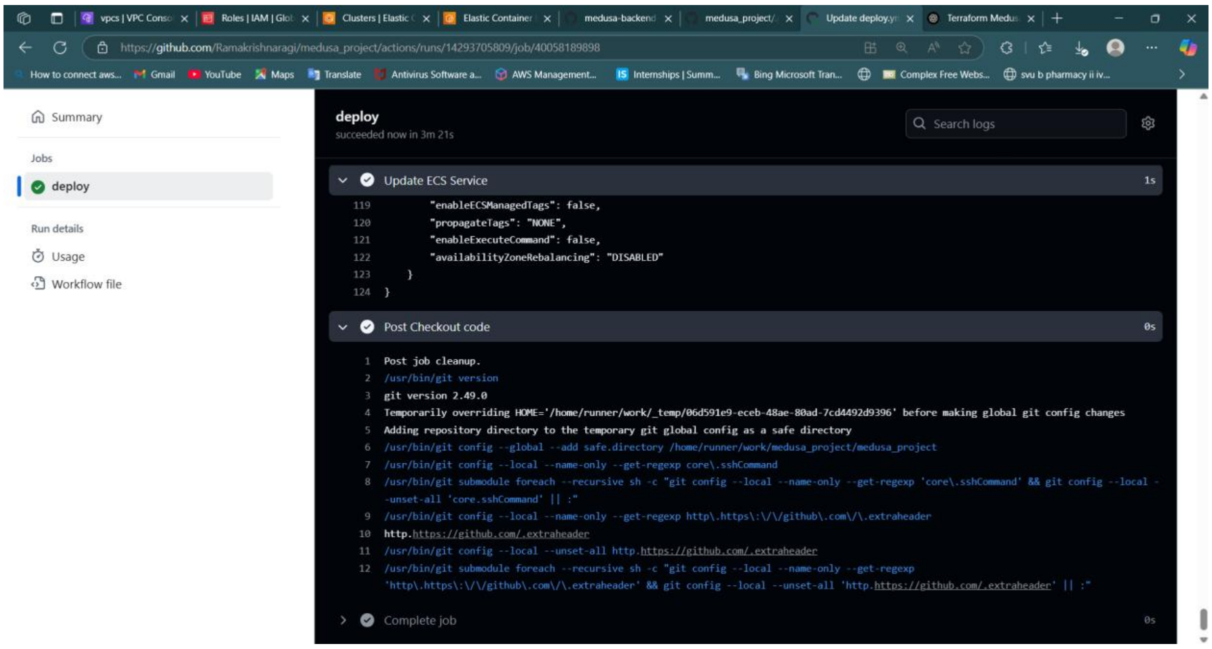
```
      run: |
        docker build -t my-image:latest -f Dockerfile .
        docker build -t medusa-backend .
        docker tag medusa-backend:latest xxxxxxxxxxxxxx.dkr.ecr.us-east-
1.amazonaws.com/medusa-backend:latest   #replace here ECR tag cmd



    - name: Push the Docker image to ECR
      run: |
        docker push xxxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/medusa-backend:latest  #
replace here ECR push cmd


    - name: Update ECS Service
      run: |
        aws ecs update-service --cluster medusa-cluster --service medusa-service --force-new-
deployment --region us-east-1
      env:
        AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY }}
        AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
        AWS_REGION: us-east-1
```

Verify the Deployment


deploy
succeeded now in 3m 21s

Update ECS Service 1s

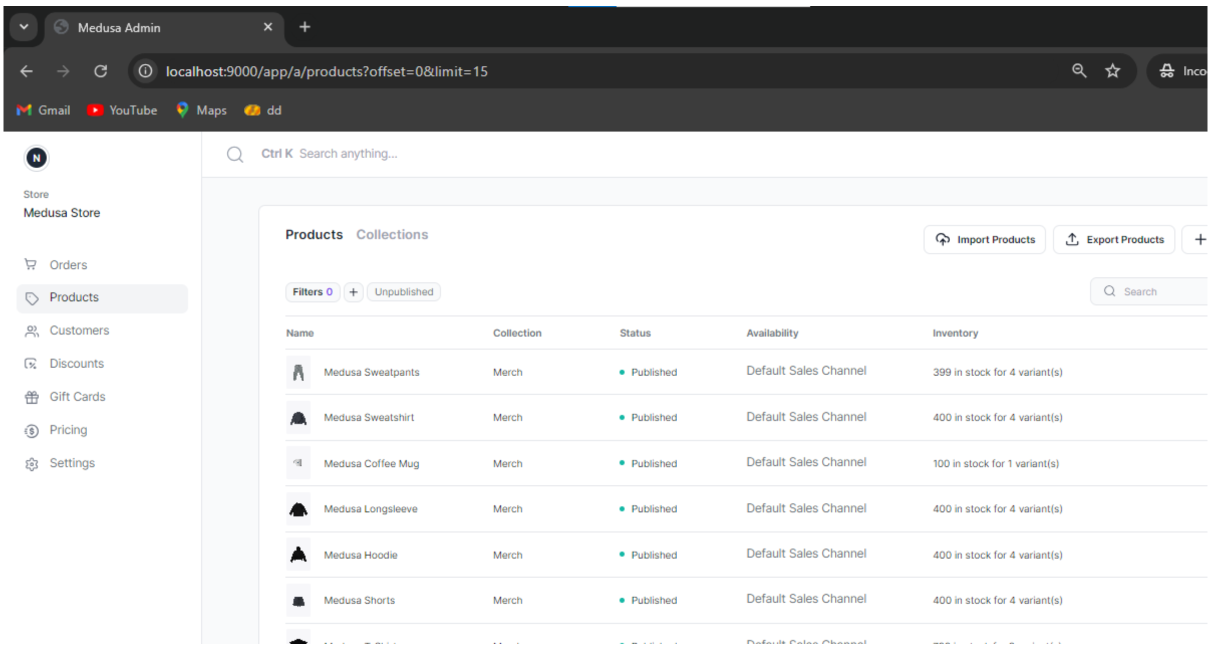
```
        "enableECSManagedTags": false,
        "propagateTags": "NONE",
        "enableExecuteCommand": false,
        "availabilityZoneRebalancing": "DISABLED"
    }
}
```

Post Checkout code 0s

```
Post job cleanup.
/usr/bin/git version
git version 2.49.0
Temporarily overriding HOME='/home/runner/work/_temp/06d591e9-eceb-48ae-80ad-7cd4492d9396' before making global git config changes
Adding repository directory to the temporary git global config as a safe directory
/usr/bin/git config --global --add safe.directory /home/runner/work/medusa_project/medusa_project
/usr/bin/git config --local --name-only --get-regexp core\.sshCommand
/usr/bin/git submodule foreach --recursive sh -c "git config --local --name-only --get-regexp 'core\.sshCommand' && git config --local -
-unset-all 'core.sshCommand' || :"
/usr/bin/git config --local --name-only --get-regexp http\.https\:\/\/github\.com\/\.extraheader
http.https://github.com/.extraheader
/usr/bin/git config --local --unset-all http.https://github.com/.extraheader
/usr/bin/git submodule foreach --recursive sh -c "git config --local --name-only --get-regexp
'http\.https\:\/\/github\.com\/\.extraheader' && git config --local --unset-all 'http.https://github.com/.extraheader' || :"
```

Complete job 0s

Once the deployment is complete, verify the resources in your AWS Management Console. Check the ECS service and task definitions to ensure everything is running correctly (please be patient medusa server will take time to start).once you saw the below image you ensure server is started properly.


Medusa Store

Products | Collections

Name	Collection	Status	Availability	Inventory
Medusa Sweatpants	Merch	Published	Default Sales Channel	399 in stock for 4 variant(s)
Medusa Sweatshirt	Merch	Published	Default Sales Channel	400 in stock for 4 variant(s)
Medusa Coffee Mug	Merch	Published	Default Sales Channel	100 in stock for 1 variant(s)
Medusa Longsleeve	Merch	Published	Default Sales Channel	400 in stock for 4 variant(s)
Medusa Hoodie	Merch	Published	Default Sales Channel	400 in stock for 4 variant(s)
Medusa Shorts	Merch	Published	Default Sales Channel	400 in stock for 4 variant(s)

Access the Medusa Backend

After verifying that your ECS service is up and running, access the Medusa backend using the public IP address or DNS name assigned to your service.

example: http://<ip-address>:9000/app/

Conclusion

In this tutorial, we successfully deployed a Medusa backend server on Amazon ECS using Terraform, demonstrating the power of Infrastructure as Code (IaC) for managing cloud resources efficiently. By following these steps, you can automate your deployments, manage configurations easily, and ensure that your infrastructure is reproducible and scalable.