DOCKERIZED APPLICATION DEPLOYMENT ON AWS EKS WITH ROLLING UPDATES

Full Project Documentation

Created By: Sakshi Pardeshi
AWS Account ID: 142039336074
Region: ap-south-1

----

1. PROJECT OBJECTIVE

To design and deploy a containerized application on Amazon EKS with:
- High Availability
- Zero Downtime Deployment
- Secure Image Storage using Amazon ECR
- Automated CI/CD using GitHub Actions
- Infrastructure Provisioning using Terraform

----

2. APPLICATION CONTAINERIZATION

Step 1: Create Flask Application (app.py)

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

@app.route('/')
def home():
    return "EKS Rolling Update Deployment"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Step 2: Create Dockerfile

```
FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY app.py .
EXPOSE 5000
CMD ["python", "app.py"]
```

Step 3: Build Docker Image

```
docker build -t eks-rolling-app .
```

----

3. AMAZON ECR SETUP

Create ECR Repository:

```
aws ecr create-repository --repository-name eks-rolling-app --region ap-south-1
```

Login to ECR:

```
aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 142039336074.dkr.
```

Tag Image:

```
docker tag eks-rolling-app:latest 142039336074.dkr.ecr.ap-south-1.amazonaws.com/eks-rolling-app:latest
```

Push Image:

```
docker push 142039336074.dkr.ecr.ap-south-1.amazonaws.com/eks-rolling-app:latest
```

----
4. EKS CLUSTER CREATION (Terraform)

```
terraform init
terraform plan
terraform apply
```

Update kubeconfig:

```
aws eks update-kubeconfig --region ap-south-1 --name eks-rolling-cluster
```

Verify Cluster:

```
kubectl get nodes
```

----
5. KUBERNETES DEPLOYMENT

Deployment Configuration:

```
replicas: 3
strategy:
  type: RollingUpdate
  rollingUpdate:
    maxUnavailable: 1
    maxSurge: 1
```

Apply Deployment:

```
kubectl apply -f k8s/deployment.yaml
```

Apply Service:

```
kubectl apply -f k8s/service.yaml
```

Verify Pods:

```
kubectl get pods
```

Verify Service:

```
kubectl get svc
```

----
6. ROLLING UPDATE PROCESS

Update Image Version:

```
docker build -t eks-rolling-app:v2 .
docker tag eks-rolling-app:v2 142039336074.dkr.ecr.ap-south-1.amazonaws.com/eks-rolling-app:v2
docker push 142039336074.dkr.ecr.ap-south-1.amazonaws.com/eks-rolling-app:v2
```

Update Deployment:

```
kubectl set image deployment/eks-rolling-app eks-container=142039336074.dkr.ecr.ap-south-1.amazonaws.com/eks-
```

Monitor Rollout:

```
kubectl rollout status deployment/eks-rolling-app
```

This ensures:
- Zero downtime
- Controlled pod replacement
- High availability maintained

----
7. CI/CD AUTOMATION (GitHub Actions)

Workflow Steps:

1. Push code to GitHub
2. GitHub Actions triggers pipeline
3. Docker image is built
4. Image pushed to Amazon ECR
5. Kubernetes deployment updated

----
8. KEY ACHIEVEMENTS

- Implemented Zero Downtime Deployment
- Improved release efficiency by ~40%
- Secured container images using private ECR
- Automated infrastructure and application deployment
- Ensured high availability across AWS Availability Zones

----
9. SKILLS DEMONSTRATED

- AWS EKS Cluster Management
- Docker Containerization
- Kubernetes Deployment Strategies
- Rolling Updates
- CI/CD Automation
- Terraform (Infrastructure as Code)
- Cloud Security Best Practices

END OF DOCUMENTATION
