**Job-Ready Real-time Project**

**Global Traffic Management**

***with***

**Multi-Region EKS Clusters**

This project deploys a simple Nginx web application that serves "**Learn IT with Azizul**" in both EKS clusters (**US**-**West** and **EU**-**West**). We will utilize a simple Node.js application for this purpose. Below, I’ll provide the steps for modifying the deployment to display your website content.

**Step 1: Create a Simple Node.js Application**

1. Create a new **directory** for your project:

**mkdir** learn-it-azizul

**cd** learn-it-azizul

2. Initialize a new Node.js project:

**npm init -y**

3. Install Express framework:

**npm install express**

4. Create an `index.js` file:

**vim** index.js

const express = require('express');

const app = express();

const PORT = process.env.PORT || **3000**;

app.get('/', (req, res) => {

res.send('<h1>**Welcome to Learn IT with Azizul!**</h1><p>This is a dynamic website served from EKS.</p>');

});

app.listen(PORT, () => {

console.log(`Server is running on port ${PORT}`);

});

5. Create a `Dockerfile` to containerize the application:

**vim D**ockerfile

# Dockerfile

**FROM node:16**

**WORKDIR /usr/src/app**

**COPY package\*.json ./**

**RUN npm install**

**COPY . .**

**EXPOSE 3000**

**CMD [ "node", "index.js" ]**

6. Build the Docker image:

**docker build -t** learn-it-azizul **.**

7. Test the Docker image locally (optional):

**docker run -p 3000:3000** learn-it-azizul

Access `http://localhost:3000` in your browser to verify the output.

**Step 2: Push Docker Image to Amazon ECR**

1. Login to Amazon ECR:

**aws ecr get-login-password --region us-west-2 | docker login --username AWS --password-stdin <account-id>.dkr.ecr.us-west-2.amazonaws.com**

2. Create an ECR repository:

**aws ecr create-repository --repository-name** learn-it-azizul --region **us**-west-2

3. Tag and push the Docker image:

**docker tag** learn-it-azizul:latest <account-id>.dkr.ecr.us-west-2.amazonaws.com/learn-it-azizul:latest

**docker push** <account-id>.dkr.ecr.us-west-2.amazonaws.com/learn-it-azizul:latest

4. Repeat the above steps to push the image to ECR in the `**eu**-west-1` region:

- **Change** the region in the `aws ecr create-repository` and `docker login` commands accordingly.

**Step 3: Deploy the Application to EKS Clusters**

**3.1 Deploy to US-West Cluster**

1. Switch context to US-West cluster:

**aws eks --region** us-west-2 **update-kubeconfig** --name eks-us-west

2. **Create** a deployment for the application:

**vim** learn-it-azizul-deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: learn-it-azizul

spec:

replicas: **2**

selector:

matchLabels:

app: **learn-it-azizul**

template:

metadata:

labels:

app: learn-it-azizul

spec:

containers:

- name: learn-it-azizul

image: <account-id>.dkr.ecr.us-west-2.amazonaws.com/learn-it-azizul:latest

ports:

- containerPort: **3000**

3. **Apply** the deployment:

**kubectl apply -f** learn-it-azizul-deployment.yaml

4. **Expose** the deployment as a service:

**vim**  learn-it-azizul-service.yaml

apiVersion: v1

kind: **Service**

metadata:

name: learn-it-azizul

spec:

type: **LoadBalancer**

ports:

- port: 80

targetPort: **3000**

selector:

app: learn-it-azizul

5. Apply the service:

**kubectl apply -f** learn-it-azizul-service.yaml

6. Get the **external** IP:

**kubectl get services**

**3.2 Deploy to Europe-West Cluster**

1. Switch context to EU-West cluster:

**aws eks --region** eu-west-1 update-kubeconfig --name eks-eu-west

2. Repeat the deployment steps (use the same YAML files) to deploy the application to the EU-West cluster.

3. Get the external IP for the EU-West service:

**kubectl get services**

**Step 4: Configure AWS Global Accelerator**

Follow the steps from the previous project to configure AWS Global Accelerator, using the Load Balancer DNS from both clusters.

**Step 5: Test the Application**

1. Use a **VPN** to simulate traffic from different locations and access the Global Accelerator DNS.

2. Check the responses to ensure that the content "Welcome to Learn IT with Azizul!" is served correctly from **both** regions.

**Step 6: Clean Up Resources**

Follow the cleanup steps provided in the previous project to delete Global Accelerator and EKS clusters.

**Conclusion**

In this project, you've successfully deployed a **dynamic** web application, "Learn IT with Azizul," across multiple EKS clusters in different AWS regions and managed traffic routing using AWS Global **Accelerator**. This project enhances your understanding of cloud architecture, Kubernetes deployments, and global traffic management.