Melvin Tejada's AI Portfolio
Website: www.melvin-ai-portfolio.com
Email: tejada.melvin@gmail.com
LinkedIn: linkedin.com/in/melvin-tejada

Deployment Documentation

Introduction

This document provides step-by-step instructions for deploying a two-part neural network deep learning model with a multi-cloud Terraform infrastructure setup. The solution is designed to deploy containers using Docker, manage infrastructure with Terraform, and configure Nginx as a web server to serve the portfolio application. Security best practices are also included to ensure the deployment is secure.

Prerequisites

Before starting the deployment, ensure the following prerequisites are met:

- **Terraform**: Version 1.10.5 or higher installed on your system.
- **Docker**: Docker engine and Docker Compose installed for container management.
- **AWS CLI**: Configured with appropriate permissions to create and manage resources.
- **Access to GitHub**: For accessing code and repository.

Steps for Deployment

Step 1: Setting up the Terraform Infrastructure

Clone the repository: Clone the repository to your local machine if you haven't already:
git clone https://github.com/tm8203/melvin-ai-portfolio.git
cd melvin-ai-portfolio

```bash
git clone https://github.com/tm8203/melvin-ai-portfolio.git
cd melvin-ai-portfolio
```

Configure AWS credentials: Ensure that your AWS credentials are configured:

```bash
aws configure
```

- **Deploy the infrastructure**:
  - Navigate to the `terraform` directory and initialize Terraform:

```bash
cd deployment/terraform
terraform init
```

- Apply the Terraform plan to provision the infrastructure:

```bash
terraform apply
```

Step 2: Docker and Nginx Setup

- **Install Docker**:
  - Make sure Docker is installed and running. If not, install Docker:

```bash
sudo apt-get install docker.io
sudo systemctl start docker
sudo systemctl enable docker
```

- **Create a `docker-compose.yml`**: In the `ml-deployment` directory, create a `docker-compose.yml` file:

Melvin Tejada's AI Portfolio
Website: www.melvin-ai-portfolio.com
Email: tejada.melvin@gmail.com
LinkedIn: linkedin.com/in/melvin-tejada

```yaml
version: '3'
services:
  web:
    image: nginx:latest
    ports:
      - "80:80"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
```

- Run the Docker containers:

```bash
docker-compose up -d
```

Step 3: Configure the Application

- **Deploy the model**: Copy the deep learning model files to the `nginx` directory and ensure your application is accessible via the correct endpoints.
- **Modify `index.html`** to include a description of your demo, the links to the video, and any documentation files.

Step 4: Testing the Deployment

Verify access: Open a browser and navigate to the public IP address of your EC2 instance:

```bash
curl -I http://<your-ec2-public-ip>
```

Step 5: Troubleshooting

- **Issue: Docker container fails to start**
  - Check if port 80 is being used by another service:

Melvin Tejada's AI Portfolio
Website: www.melvin-ai-portfolio.com
Email: tejada.melvin@gmail.com
LinkedIn: linkedin.com/in/melvin-tejada

```bash
sudo lsof -i :80
```

    - Stop the conflicting service and try again.
- **Issue: Permissions errors during Terraform apply**
    - Ensure that the IAM role you are using has the necessary permissions to create resources like EC2 instances, security groups, and IAM roles.

Conclusion

Once you've followed these steps, your multi-cloud deployment should be live. The application will be accessible through an Nginx reverse proxy, with your model running securely in Docker containers. The next steps include testing the infrastructure, scaling if necessary, and continuing to monitor the deployed solution.