Deploy an EKS Cluster with Terraform

Project Overview

This project uses **Terraform** to provision an **Amazon EKS Cluster** on AWS. By leveraging **Infrastructure as Code (IaC)**, we automate the deployment of Kubernetes clusters with modular and reusable Terraform configurations.

◇ **Kubernetes (K8s)** manages containerized applications efficiently. ◇ **EKS (Elastic Kubernetes Service)** is a managed K8s solution on AWS. ◇ **Terraform** simplifies infrastructure provisioning with reusable modules.

With this setup, you can deploy, manage, and scale Kubernetes workloads seamlessly!

📁 Project Structure

The repository follows a **modular structure** for better organization and reusability:

```
EKS-CLUSTER-TERRAFORM/
│── modules/                  # Terraform modules
│   ├── eks/                  # EKS module
│   │   ├── main.tf           # Defines EKS cluster
│   │   ├── outputs.tf        # Outputs for EKS cluster
│   │   ├── variables.tf      # Variables for EKS cluster
│   ├── vpc/                  # VPC module
│   │   ├── main.tf           # Defines networking resources
│   │   ├── outputs.tf        # Outputs for VPC
│   │   ├── variables.tf      # Variables for VPC
│
│── .gitignore                # Git ignore file
│── LICENSE                   # License file
│── kubectl.sha256            # Checksum for kubectl
│── .terraform.lock.hcl        # Terraform lock file
│── kubernetes.tf             # Kubernetes resources definition
│── main.tf                   # Root Terraform configuration
│── outputs.tf                 # Root outputs
│── provider.tf                 # Provider configurations
│── README.md                   # Documentation (this file)
```

◇ `modules/eks` – Manages EKS cluster deployment. ◇ `modules/vpc` – Handles VPC and networking setup. ◇ `provider.tf` – Defines the AWS provider. ◇ `main.tf` – Root Terraform script to call modules. ◇ `outputs.tf` – Stores and displays useful deployment details.

⚡ Prerequisites

- ☑ **AWS Account** – Sign up at AWS if you don't have one.
- ☑ **Terraform** – Install from Terraform's official site.
- ☑ **AWS CLI** – Install and configure credentials (guide).
- ☑ **kubectl** – Kubernetes CLI tool (installation guide).
- ☑ **VS Code (Optional)** – Recommended IDE for managing Terraform code.

🚀 Deployment Steps

1 Clone the Repository

```
git clone https://github.com/devops-practicals/eks-terraform
cd eks-terraform
```

---

2 Terraform Init, Plan, Apply

Run the following Terraform commands:

```
terraform init      # Initialize Terraform backend
terraform plan      # Preview infrastructure changes
terraform apply     # Deploy infrastructure
terraform destroy   # Destroy infrastructure
```

📌 **Expected Outputs:**

- EKS Cluster Name
- Node IP Addresses
- VPC ID

---

3 Connect to Your EKS Cluster

After deployment, retrieve cluster credentials:

```
AWS_REGION="ap-south-1"
EKS_CLUSTER_NAME="terraform-eks-test-cluster"
aws eks update-kubeconfig --name $EKS_CLUSTER_NAME --region $AWS_REGION
```

Verify cluster connectivity:

```
kubectl get nodes
```

4️⃣ Terraform S3 Backend Integration (Optional)

If you want to integrate with **S3 Backend**:

1️⃣ Update the **backends.tf** with bucket,key. 2️⃣ Create a new **Workspace** and connect your GitHub repository. 3️⃣ Add the following environment variables:

```
AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY
AWS_DEFAULT_REGION
CONFIRM_DESTROY
```

5️⃣ Destroy the Infrastructure

To **delete the deployed resources**, run:

```
terraform destroy
```
