**Documentation For Deploy the Web Application on a Cloud-Based Kubernetes Cluster**

This document outlines the steps to deploy a containerized web application in a cloud-based Kubernetes cluster using Terraform for infrastructure provisioning and Prometheus for monitoring. Follow the steps below to set up the solution end-to-end.

**Prerequisites**

Ensure you have the following tools installed on your local machine:

* **AWS CLI** or the appropriate cloud provider
* **kubectl**: Kubernetes command-line tool
* **Terraform**: Infrastructure-as-Code tool
* **Docker**: For building and pushing container images

**Step 1: Provision Cloud Infrastructure using Terraform**

Terraform will be used to provision the necessary cloud infrastructure (such as Kubernetes cluster, VPC, subnets, security groups).

**Steps:**

1. **Navigate to the Terraform directory:** Go to the directory where the Terraform configuration files are located.

cd D:\cloud-devops-task

1. **Initialize Terraform:** Initialize the working directory with the required provider plugins.

terraform init

1. **Terraform Plan:** Generate and review the execution plan to verify which resources will be created.

terraform plan

1. **Apply Terraform Configuration:** Create the cloud infrastructure (Kubernetes cluster, networking components) by running:

terraform apply

1. **Get Kubernetes Cluster Configuration:** Once the cluster is created, use the cloud provider's CLI (e.g., AWS) to configure kubectl to communicate with the cluster.

For AWS EKS:

aws eks --region ap-south-1 update-kubeconfig --name web-app-cluster

This will update your kubeconfig file to connect kubectl to the newly created Kubernetes cluster.

**Step 2: Build and Push the Docker Image**

The web application (a static HTML page) will be containerized using Docker. You need to build and push this image to a container registry like Docker Hub or AWS ECR.

**Steps:**

1. **Navigate to your project directory:**

cd D:\cloud-devops-task

1. **Build the Docker image:**

docker build -t web-app-image .

1. **Push the Docker image to the registry:**

Login into the docker hub:

Docker login

Tag the docker image:

Docker tag web-app-image dockerhub-username/web-app-image:latest

Push the image to docker hub

Docker push username/web-app-repo:latest

**Step 3: Deploy the Application to Kubernetes**

Now that the infrastructure is ready and the Docker image has been pushed, Tag deploy the application to Kubernetes.

**Steps:**

1. **Deploy the application using Kubernetes manifests:**

Apply the deployment and service YAML files to deploy the web application and expose it.

kubectl apply -f /path-to-your-project/k8s/deployment.yaml

kubectl apply -f /path-to-your-project/k8s/service.yaml

1. **Check the status of the application:** You can monitor the status of the deployed pods by running:

kubectl get pods

1. **Get the service's external IP address:** The service should expose the web application using a LoadBalancer service. Retrieve the external IP or DNS name:

kubectl get svc static-web-app-service

1. **Access the application:** Visit the external IP address or DNS name in your browser to see the web application.

**Step 4: Set Up Monitoring with Prometheus**

To ensure proper monitoring is in place, Prometheus is configured to scrape metrics from the Kubernetes cluster.

**Steps:**

1. **Deploy Prometheus:** Apply the Prometheus configuration in the k8s/prometheus-config.yaml file.

kubectl create namespace monitoring

kubectl apply -f /d/cloud-devops-task/k8s/deployment.yaml

kubectl apply -f /d/cloud-devops-task/k8s/service.yaml

1. **Verify Prometheus deployment:** Check if the Prometheus pod is running by listing all pods in the monitoring namespace:

kubectl get pods -n monitoring

1. **Access Prometheus (Optional):** You can expose the Prometheus dashboard to view the collected metrics. You can set up a port-forward to access it locally:

kubectl port-forward -n monitoring prometheus-server-abc123 9090:9090

Access the Prometheus UI via http://localhost:9090.

**Step 5: Verify the Setup**

After completing the deployment, ensure the following:

* **Application is accessible**: Visit the external IP (LoadBalancer) and verify the static content of the web application is loading.
* **Monitoring is working**: Check Prometheus is scraping Kubernetes metrics by accessing the Prometheus dashboard (optional).