Deploying a Web application using Cloud based Kubernetes solution

Overview

In this task, I deployed a simple web application using azure kubernetes service. I used terraform to create aks cluster and monitor cluster using azure managed prometeus and grafana. I am using github codeworkspace as a working platform(server).

Step 1: Terraform automation for aks cluster creation

Install terraform and Azure cli in workspace.

Create 3 tf files (main.tf, variables.tf, outputs.tf) for aks cluster creation.

main.tf

```
 provider "azurerm" {
   features {}
# Service principal authentication
   client_id             = var.client_id
   client_secret      = var.client_secret
   tenant_id             = var.tenant_id
   subscription_id = var.subscription_id
}
 # Resource group
resource "azurerm_resource_group" "aks_rg" {
   name         = var.resource_group_name
   location = var.location
}
```

```
 # AKS Cluster
resource "azurerm_kubernetes_cluster" "aks_cluster" {
  name                = var.aks_cluster_name
  location            = azurerm_resource_group.aks_rg.location
  resource_group_name = azurerm_resource_group.aks_rg.name
  dns_prefix          = var.dns_prefix
 default_node_pool {
    name       = "default"
    node_count = var.node_count
    vm_size    = var.vm_size
  }
 identity {
    type = "SystemAssigned"
  }
}
 # Helm Provider
provider "helm" {
  kubernetes {
    config_path = "${path.module}/kubeconfig"
  }
}
```

outputs.tf

```
output "aks_cluster_name" {
  description = "The name of the AKS cluster"
  value       = azurerm_kubernetes_cluster.aks_cluster.name
}

  output "kube_config" {

    description = "Kubeconfig to access the AKS cluster"

    value             = azurerm_kubernetes_cluster.aks_cluster.kube_config_raw

    sensitive      = true

}
```

variables.tf

```
variable "location" {

    description = "Azure region for resources"

    default         = "East US"

}

variable "resource_group_name" {

    description = "Name of the Azure resource group"

    default         = "aks-resource-group"

}

  variable "aks_cluster_name" {

    description = "Name of the AKS cluster"

    default         = "aks-cluster"

}

  variable "dns_prefix" {

    description = "DNS prefix for the AKS cluster"

    default         = "aks"

}

  variable "node_count" {

    description = "Number of nodes in the AKS cluster"
    default          = 2
}
  variable "vm_size" {
    description = "VM size for AKS nodes"
    default          = "standard_a2_v2"
}
  # Service principal variables
variable "client_id" {
    description = "b73506d0-fa3f-460f-8134-bdf494d49e3a"
    sensitive      = true
}
  variable "client_secret" {
    description = "hsj8Q~QuVhjN9fdRPlUbrAlFx45VZtqgnHkp2cJl"
    sensitive      = true
}
  variable "tenant_id" {
    description = "0de5ebc3-9158-473f-ae12-61641968546b"
}
  variable "subscription_id" {
    description = "1047e392-93b4-4db3-a22b-927151ce153b"
}
```

execute terraform code by,

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

Now, In azure dashboard aks cluster have been successfully created.

Step 2: Create docker image using dockerfile and upload in my dockerhub account

create one index.html for simple web page in your workspace

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Simple Static Web Page</title>
<style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            margin-top: 50px;
        }
        h1 {
            color: #0078D7;
        }
      p {
            color: #555;
        }
</style>
</head>
<body>
<h1>Welcome to My Static Web Page</h1>
<p>This is a simple web page served from a Docker container.</p>
</body>
</html>
```

Install docker in your workspace and create dockerfile

Dockerfile

```
# Use the official Nginx image as the base image
FROM nginx:alpine
  # Copy the static web page to the Nginx default HTML directory
COPY index.html /usr/share/nginx/html/
  # Expose port 80 to access the web page
EXPOSE 80
  # Start the Nginx server
CMD ["nginx", "-g", "daemon off;"]
```

Build dockerfile and upload your image into dockerhub account

```
docker login(enter your dockerhub account username and password)
docker build -t ananthsunrise/simple-static-web .
docker push ananthsunrise/simple-static-web
```

Step 3: Create deployment files and apply using kubectl

Install kubectl in your workspace.

create 2 files (deployment.yaml, sevice.yaml)

deployment.yaml

```
apiVersion: apps/v1

kind: Deployment

metadata:

  name: static-web-deployment

  labels:

    app: static-web

spec:

  replicas: 2

  selector:

    matchLabels:

      app: static-web

  template:

    metadata:

      labels:

        app: static-web

    spec:

      containers:

      - name: static-web

        image: ananthsunrise/simple-static-web:latest

        ports:

        - containerPort: 80
```

service.yaml

```
apiVersion: v1

kind: Service

metadata:

  name: static-web-service

spec:

  selector:

    app: static-web

  ports:

  - protocol: TCP

    port: 80

    targetPort: 80

  type: LoadBalancer
```

apply files using kubectl

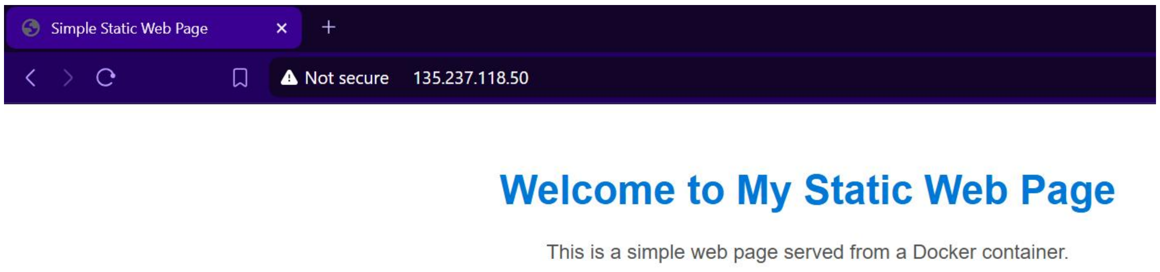
```
kubectl apply -f deployment.yaml

kubectl apply -f service.yaml
```

Step 4:Access my webpage in browser

```
kubectl get svc
```

(it will give ip of load balancer to access your webpage on port 80)



Simple Static Web Page

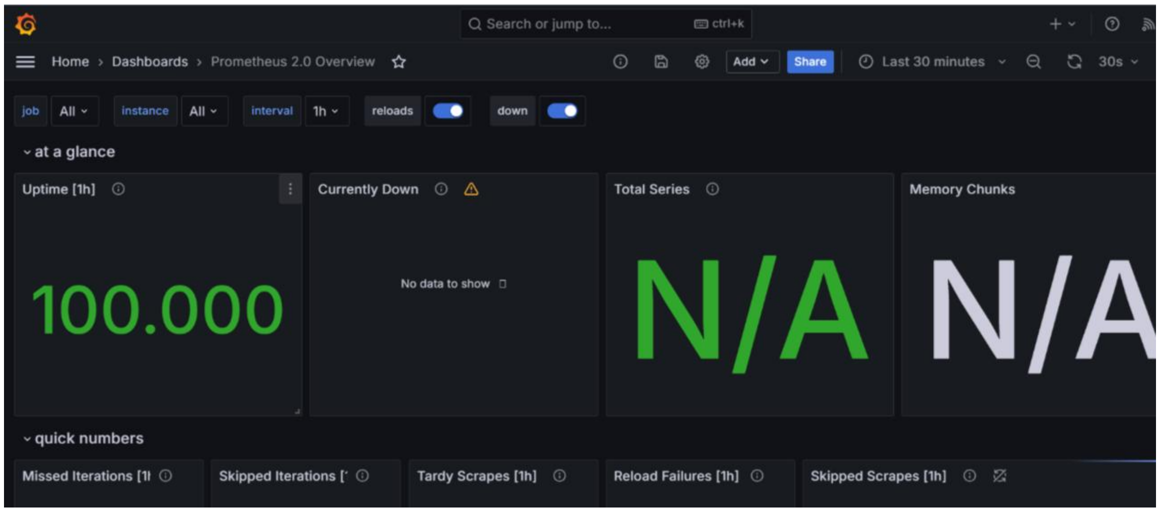
Not secure 135.237.118.50

Welcome to My Static Web Page

This is a simple web page served from a Docker container.

Step 5:Monitoring kubernetes solution using azure managed prometheus and grafana

In this i am using azure managed prometheus and grafana to monitor cluster resources.



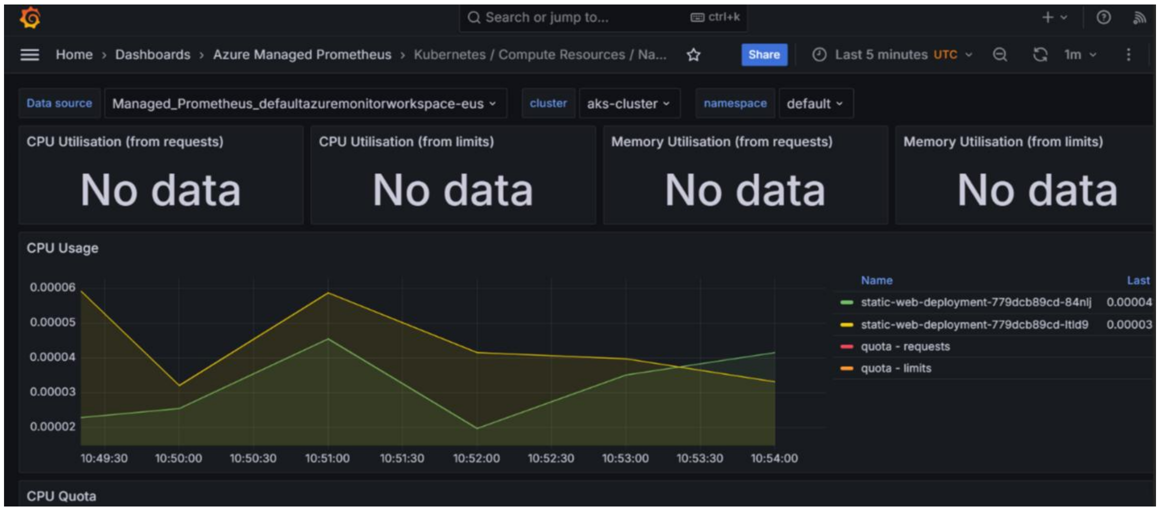
job All · instance All · interval 1h · reloads · down

at a glance

Uptime [1h]	Currently Down	Total Series	Memory Chunks
100.000	No data to show	N/A	N/A

quick numbers

Missed Iterations [1h · Skipped Iterations [' · Tardy Scrapes [1h] · Reload Failures [1h] · Skipped Scrapes [1h]



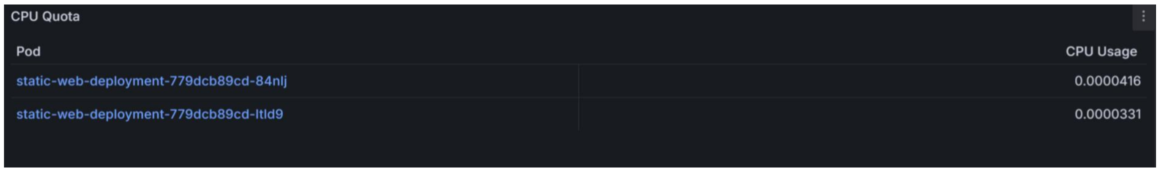
Data source Managed_Prometheus_defaultazuremonitorworkspace-eus · cluster aks-cluster · namespace default

CPU Utilisation (from requests)	CPU Utilisation (from limits)	Memory Utilisation (from requests)	Memory Utilisation (from limits)
No data	No data	No data	No data

CPU Usage

Name	Last
static-web-deployment-779dcb89cd-84nlj	0.00004
static-web-deployment-779dcb89cd-ltld9	0.00003
quota - requests	
quota - limits	

CPU Quota



CPU Quota

Pod	CPU Usage
static-web-deployment-779dcb89cd-84nlj	0.0000416
static-web-deployment-779dcb89cd-ltld9	0.0000331