Deploy AKS Using Terraform

I. Overview

- The goal of this document is to demonstrate how to deploy AKS, VNet, Prometheus, and Grafana to Azure using Terraform.
- In this demo (my repository), I leveraged the original provided repository and added some new resources to meet the assignment requirements.

II. Implement

1. Remote Backend configuration

In the top of **./main.tf** add following block

```
terraform {
  backend "azurerm" {
      resource_group_name  = "myTFResourceGroup"
      storage_account_name = "minhpraticekubernetes"
      container_name       = "tfstate"
      key                  = "terraform.tfstate"
  }

}
```

This block configures the **Terraform backend** to use **Azure Storage** for remote state management.

It stores the Terraform state file (`terraform.tfstate`) in the specified storage

account, container, and resource group, ensuring state consistency and team collaboration.

2. ACR

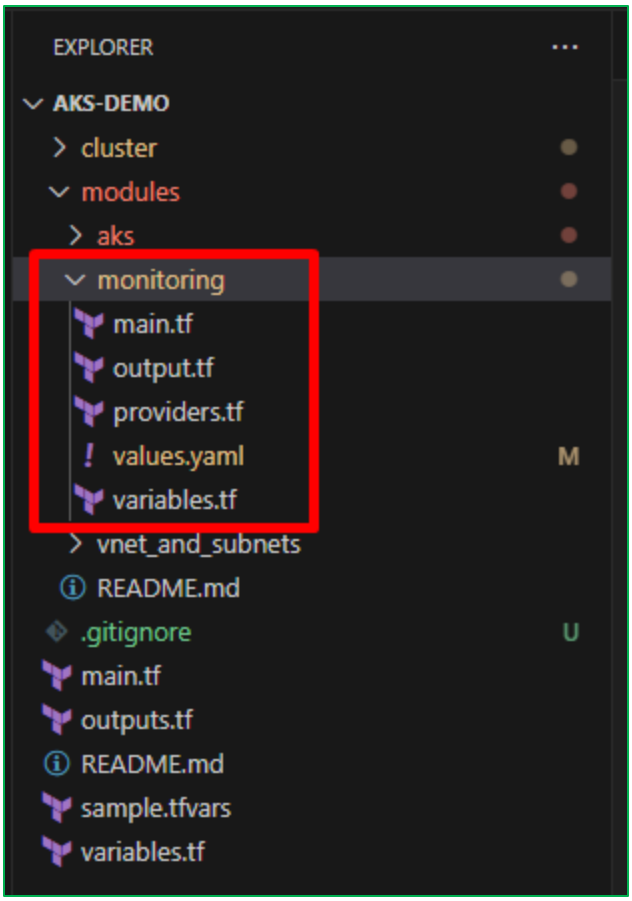
In the end of **.modules/aks/main.tf** add a block to define ACR

```
resource "azurerm_container_registry" "minhtestacr" {
  name                = "minhtestacr"
  resource_group_name = var.resource_group_name
  location            = var.region
  sku                 = "Standard"
}

resource "azurerm_role_assignment" "minh_aks_acr_pull" {
  principal_id                     =
azurerm_kubernetes_cluster.k8s.kubelet_identity[0].object_id
  role_definition_name             = "AcrPull"
  scope                            =
azurerm_container_registry.minhtestacr.id
  skip_service_principal_aad_check = true
}
```

block `azurerm_role_assignment` will create a IAM role to allow AKS pull image from ACR

3. Prometheus & Grafana addition

Add module monitoring which define code to deploy Prometheus and Grafana using helm




```hcl
# main.tf
resource "helm_release" "kube_prometheus_stack" {
  # Helm repository details
  name       = "kube-prometheus-stack"
  repository = "https://prometheus-community.github.io/helm-charts"
  chart      = "kube-prometheus-stack"
  version    = "59.0.0" # Use a stable, recent version
  namespace  = "monitoring"
  create_namespace = true

  values = [
    templatefile("${path.module}/values.yaml", {
      ip = ""
    })
  ]

  # Optional: Wait for all resources to be ready (can take a few minutes)
  timeout = 600
  wait    = true
}
```

This Helm release installs the **Kube Prometheus Stack** chart into the monitoring namespace on AKS. It deploys Prometheus, Grafana, and related monitoring components using a specified Helm chart version from the Prometheus Community repository.

```
# providers.tf
# Configure the Kubernetes Provider
provider "kubernetes" {
  host                   = var.host
  client_certificate     = base64decode(var.client_certificate)
  client_key             = base64decode(var.client_key)
  cluster_ca_certificate = base64decode(var.cluster_ca_certificate)
}

# Configure the Helm Provider
provider "helm" {
  kubernetes = {
    host                   = var.host
    client_certificate     = base64decode(var.client_certificate)
    client_key             = base64decode(var.client_key)
    cluster_ca_certificate = base64decode(var.cluster_ca_certificate)
  }
}
```

This configuration defines the **Kubernetes** and **Helm** providers used to interact with the AKS cluster.
It connects to the cluster using credentials (host, client certificate, client key, and CA certificate) passed through Terraform variables.

```
# values.yaml
prometheus:
  prometheusSpec:
    podMonitorSelectorNilUsesHelmValues: false
    serviceMonitorSelectorNilUsesHelmValues: false
    ruleSelectorNilUsesHelmValues: false


grafana:
  adminUser: admin
  adminPassword: admin
  grafana.ini:
```

```
    server:
      root_url: "%(protocol)s://%(domain)s/grafana/"
```

This **values.yaml** snippet customizes the **Kube Prometheus Stack** deployment.

- Under `prometheus.prometheusSpec`, it ensures Prometheus recognizes **PodMonitors**, **ServiceMonitors**, and **Rules** created outside of the Helm release (by disabling the default Helm-only selectors).
- The `grafana` section sets up Grafana with default admin credentials and defines a custom `root_url` for accessing the Grafana UI (then we can access Grafana via <Ingress controller IP>/grafana)

```
# variables.tf
variable "host" {
  type        = string
  description = "aks cluster host"
}

variable "client_certificate" {
  type        = string
  description = "aks cluster client certificate"
}

variable "client_key" {
  type        = string
  description = "aks cluster client key"
}

variable "cluster_ca_certificate" {
  type        = string
  description = "aks cluster cluster ca certificate"
}
```

This file defines the Terraform input variables required to authenticate and connect to the AKS cluster.

In **./modules/aks/outputs.tf** add some outputs:

```
output "client_certificate" {
  description = "Base64 encoded public certificate used by clients to
authenticate to the Kubernetes cluster."
  value       =
azurerm_kubernetes_cluster.k8s.kube_config.0.client_certificate
```

```
  sensitive   = true
}
output "host" {
  description = "Kubernetes cluster host."
  value       = azurerm_kubernetes_cluster.k8s.kube_config.0.host
  sensitive   = true
}

output "client_key" {
  description = "Base64 encoded public client_key."
  value       = azurerm_kubernetes_cluster.k8s.kube_config.0.client_key
  sensitive   = true
}

output "cluster_ca_certificate" {
  description = "Base64 encoded public cluster_ca_certificate."
  value       =
azurerm_kubernetes_cluster.k8s.kube_config.0.cluster_ca_certificate
  sensitive   = true
}
```

Finally, in **./cluster/main.tf** add section to call the monitoring module and pass value for variables

```
module "monitoring" {
  # invoke monitoring module under modules directory
  source = "../modules/monitoring"

  host = module.aks_with_node_group.host
  client_certificate=module.aks_with_node_group.client_certificate
  client_key=module.aks_with_node_group.client_key
  cluster_ca_certificate=module.aks_with_node_group.cluster_ca_certificate

}
```

4. Execution

Cd to root of repository folder, execute

terraform init

terraform apply -var-file="sample.tfvars"

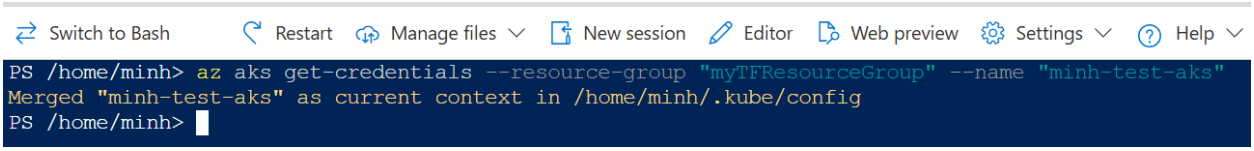
and then enter “yes” to confirm start process

```
PS /home/minh> cd ./aks-demo/
PS /home/minh/aks-demo> terraform init
Initializing the backend...
Initializing modules...
Initializing provider plugins...
- Reusing previous version of hashicorp/helm from the dependency lock file
- Reusing previous version of hashicorp/kubernetes from the dependency lock file
- Reusing previous version of hashicorp/azurerm from the dependency lock file
- Using previously-installed hashicorp/helm v3.1.0
- Using previously-installed hashicorp/kubernetes v2.38.0
- Using previously-installed hashicorp/azurerm v3.107.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS /home/minh/aks-demo> terraform apply -var-file="sample.tfvars"
Acquiring state lock. This may take a few moments...
module.cluster.module.vnet_with_subnets.azurerm_virtual_network.minh_az_vnet: Refresh
f55f2a88/resourceGroups/myTFResourceGroup/providers/Microsoft.Network/virtualNetworks
```

5. Verify deployed resource

Use az cli command to access AKS

az aks get-credentials --resource-group "your_resource_group" --name "your_AKS"



Use kubectl get all -n monitoring to verify Prometheus and Grafana

```
PS /home/minh> kubectl get all -n monitoring
NAME                                                     READY   STATUS    RESTARTS   AGE
pod/alertmanager-kube-prometheus-stack-alertmanager-0    2/2     Running   0          4h29m
pod/kube-prometheus-stack-grafana-7d47bc94b9-g9q2s       3/3     Running   0          3h18m
pod/kube-prometheus-stack-kube-state-metrics-5b8c8b74d5-rgxxl   1/1     Running   0          4h29m
pod/kube-prometheus-stack-operator-768986f7c5-881jb      1/1     Running   0          4h29m
pod/kube-prometheus-stack-prometheus-node-exporter-72rjz 1/1     Running   0          4h29m
pod/kube-prometheus-stack-prometheus-node-exporter-ptxz2 1/1     Running   0          4h29m
pod/prometheus-kube-prometheus-stack-prometheus-0        2/2     Running   0          4h29m

NAME                                              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                      AGE
service/alertmanager-operated                     ClusterIP   None            <none>        9093/TCP,9094/TCP,9094/UDP   4h29m
service/kube-prometheus-stack-alertmanager        ClusterIP   10.0.74.156     <none>        9093/TCP,8080/TCP            4h29m
service/kube-prometheus-stack-grafana             ClusterIP   10.0.161.193    <none>        80/TCP                       4h29m
service/kube-prometheus-stack-kube-state-metrics  ClusterIP   10.0.151.239    <none>        8080/TCP                     4h29m
service/kube-prometheus-stack-operator            ClusterIP   10.0.39.166     <none>        443/TCP                      4h29m
service/kube-prometheus-stack-prometheus          ClusterIP   10.0.146.192    <none>        9090/TCP,8080/TCP            4h29m
service/kube-prometheus-stack-prometheus-node-exporter   ClusterIP   10.0.204.239    <none>        9100/TCP                     4h29m
service/prometheus-operated                       ClusterIP   None            <none>        9090/TCP                     4h29m

NAME                                                    DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR
daemonset.apps/kube-prometheus-stack-prometheus-node-exporter   2         2         2       2            2           kubernetes.io/os=linux
9m

NAME                                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/kube-prometheus-stack-grafana       1/1     1            1           4h29m
deployment.apps/kube-prometheus-stack-kube-state-metrics   1/1     1            1           4h29m
deployment.apps/kube-prometheus-stack-operator      1/1     1            1           4h29m

NAME                                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/kube-prometheus-stack-grafana-544c99459        0         0         0       4h29m
replicaset.apps/kube-prometheus-stack-grafana-7d47bc94b9       1         1         1       3h18m
replicaset.apps/kube-prometheus-stack-kube-state-metrics-5b8c8b74d5   1         1         1       4h29m
replicaset.apps/kube-prometheus-stack-operator-768986f7c5      1         1         1       4h29m
```