Caso Práctico 2

Automatización de despliegues en entornos Cloud

**Irene González Cantera**

**Entregables**

1. La URL del repositorio git donde está disponible el código con la resolución de este caso práctico.

Repositorio: azure-cp2-repo

URL: <https://github.com/irenegcantera/azure-cp2-repo.git>

1. Diagramas mostrando los elementos desplegados y su rol. Estos diagramas deberán ser realizados por el alumno.

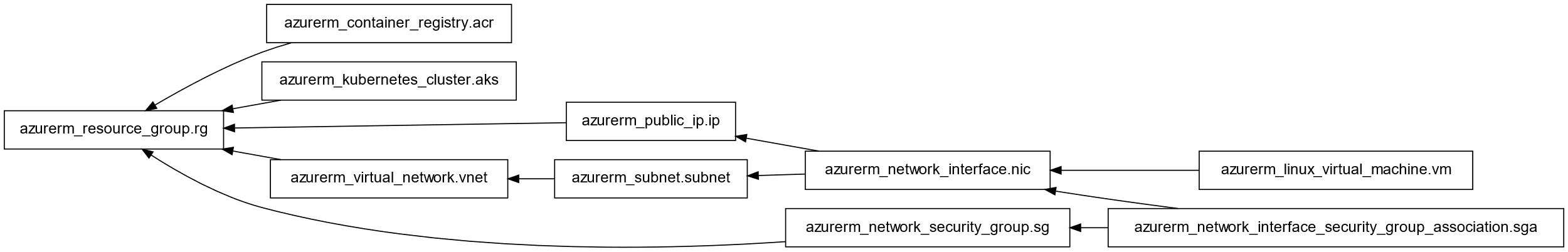
He generado los dos siguientes diagramas con la herramienta ***terraform graph***. Para hacer uso de esta herramienta he instalado Graphviz.

***sudo apt-get install graphviz***

Después he lanzado estos comandos para generar los dos diagramas (se encuentran en la carptea **/charts** para verlos mejor ya que el segundo no se ve con calidad):

***terraform graph | dot -Tsvg > ../charts/17032024.svg***

En este diagrama se ve una visión general de los recursos creados en Azure.



***terraform graph -type=plan | dot -Tpng > ../charts/17032024.png***

En este diagrama se ve una visión más completa donde se relaciona todos los componentes de la configuración, es decir, todas las relaciones entre ficheros, aparecen las variables inputs y outputs y los recursos.

Diagrama

Descripción generada automáticamente

En el directorio **/charts** tenemos los dos diagramas:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. Descripción del proceso de despliegue, sus requisitos y los recursos resultantes en Azure.

Para crear la infraestructura en el entorno cloud de Azure se va a utilizar Terraform. Las infraestructuras que se van a crear son Azure Container Registry (ACR), Virtual Machine (VM) y Azure Kubernetes Service (AKS).

En el directorio **/terraform** tenemos los siguientes ficheros:

|  |  |
| --- | --- |
| Interfaz de usuario gráfica, Aplicación  Descripción generada automáticamente |  |

En provider.tf definimos el proveedor cloud y la versión. Para la práctica usaremos el proveedor de Azure y la version 3.85.0.

En network.tf se crea la red para la máquina virtual que se creará más adelante.

En outputs.tf guardaremos resultados como contraseñas, la IP pública o valores que se utilizarán en otro momento.

En recursos.tf se define la creación de un grupo de recursos donde se asociarán otros recursos, es decir, en ese grupo encontraremos asociado el recurso de máquina virtual, ACR y AKS.

En security.tf se asocia diferentes reglas de seguridad de acceso a la red creada anteriormente. Definimos que tenga como reglas la conexión por SSH y el acceso a través de HTTP y HTTPS.

En vars.tf se almacena variables que se utilizan en los anteriores archivos.

Antes de desplegar la infraestructura hay que iniciar sesión en el proveedor con Azure CLI y guardamos el ID de la subscrición para crear un Service Principal.

***az ad sp create-for-rbac --role="Contributor"***

***--scopes="/subscriptions/<SUBSCRIPTION\_ID>"***

Después de crear ese Service Principal obtendremos unas credenciales que serán usadas por Terraform para conectarse a Azure. Añadimos esas credenciales como variables de entorno:

***export ARM\_CLIENT\_ID="<APPID\_VALUE>"***

***export ARM\_CLIENT\_SECRET="<PASSWORD\_VALUE>"***

***export ARM\_SUBSCRIPTION\_ID="<SUBSCRIPTION\_ID>"***

***export ARM\_TENANT\_ID="<TENANT\_VALUE>"***

Creamos la clave pública para poder realizar las conexiones por SSH entre la máquina virtual y otros recursos.

A continuación, comenzamos con el despliegue.

Inicializamos un directorio de trabajo que contiene los archivos de configuración de Terraform.

***terraform init***

Comprobamos que los archivos de configuración presentan el formato correcto y si la configuración es válida.

***terraform fmt***

***terraform validate***

Creamos un plan de ejecución para previsualizar los cambios que Terraform planea realizar en su infraestructura.

***terraform plan -out <plan\_name>.tfplan***

Ejecutamos las acciones propuestas en el plan de ejecución.

***terraform apply <plan\_name>.tfplan***

Ejecutamos tras unos minutos el siguiente comando, ya que la IP pública puede tardar en asignarse y aparecer vacío en las variables de los outputs.

***terraform refresh***

Destruimos todos los objetos remotos gestionados una vez terminemos con Azure y evitar costes innecesarios.

***terraform destroy***

Los recursos resultantes son los siguientes:

Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Descripción de la aplicación desplegada.

Para hacer el despliegue de Apache en la máquina virtual se ha usado Ansible, por lo que se ha desarrollado diferentes archivos YAML para realizar el despliegue.

En el directorio **/ansible** del repositorio git tenemos los siguientes ficheros:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ejecutamos el playbook setup.yml para instalar Podman en la máquina virtual de Azure.

**ansible-playbook -i inventory.yml setup.yml**

Texto

Descripción generada automáticamente con confianza media

Ejecutamos el playbook push\_images\_registry.yml para descargar la imagen Apache y la base de datos MySQL. Las imágenes se subirán al contendor registry creado en Azure con el tag casopractico2.

**ansible-playbook -i inventory.yml push\_images\_registry.yml**

Imagen que contiene Escala de tiempo

Descripción generada automáticamente

Ambas imágenes se subirán en la ruta cp2 del repositorio de ACR, pero la imagen de Apache se localizará en cp2/podman/httpd y la imagen de MySQL en cp2/aks/mysql.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ejecutamos el playbook deploy\_container\_podman.yml para descagar la imagen subida con anterioridad al ACR y desplegar el contenedor de Apache server en la máquina virtual en el puerto 80.

**ansible-playbook -i inventory.yml deploy\_container\_podman.yml**

Escala de tiempo

Descripción generada automáticamente

Si accedemos a la dirección web con el puerto 80, veremos que Apache se encuentra levantado correctamente.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Despliegue de la aplicación en AKS no realizado.**

1. Descripción de los problemas encontrados, si los hubiera, cómo se han solucionado y referencias que se hayan utilizado para resolverlas (en formato APA). Si algún problema no se ha podido solucionar, se deberá analizar y proponer soluciones o siguientes pasos.

Me he encontrado con los siguientes errores:

1. Texto

   Descripción generada automáticamenteAl lanzar el comando ***terraform validate*** para validar el formato de mis archivos TF, me devolvió error de que no encontraba el fichero id\_rsa.pub en la ruta indicada.

No era consciente de que no existía el fichero, por lo que para solucionarlo he generado las claves para conexiones SSH con el comando ***ssh-keygen -t rsa***. Una vez realizado esa acción terraform indica que es válida mi configuración.

1. Interfaz de usuario gráfica, Texto

   Descripción generada automáticamenteEl siguiente error que me daba era a la hora de crear el recurso cluster de Kubernetes en AKS. Este error indicaba que la versión de la api no era soportada.

Para solucionar el error consulté en el foro de discusión del curso y se me sugirió que se cambiase la versión del provider a una más reciente. Esta sugerencia se debe a que la versión que estaba utilizando estaba deprecada.

En mi archivo de configuración provider.tf tenía puesto lo siguiente:

terraform {  required\_providers {    azurerm = {      source  = "hashicorp/azurerm"      version = "~> 3.0.2"    }  }

  required\_version = ">= 1.1.0"}

Y modifiqué a las siguientes versiones:

terraform {

required\_providers {

azurerm = {

source = "hashicorp/azurerm"

version = "3.85.0"

}

}

required\_version = ">= 0.14"

}

1. Texto

   Descripción generada automáticamenteA raíz del cambio de la versión el comando terraform init fallaba porque no encontraba la versión almacenada anteriormente en el fichero de configuración provider.tf. Por tanto, había que hacer una actualización de los plugins con ***terraform init -upgrade***
2. Tuve problemas al lanzar los commandos de ***ansible-playbook***. No conseguía la conexión de Ansible con mi máquina virtual en Azure.

Error:

fatal: [40.125.70.164]: UNREACHABLE! => {"changed": false, "msg": "Failed to connect to the host via ssh: irene.gonzalez751@comunidadunir.net@40.125.70.164: Permission denied (publickey).", "unreachable": true}

Imagen de la pantalla de una computadora

Descripción generada automáticamente con confianza baja

Al principio en mi configuración de la máquina virtual tenía puesto que me conectaba con la public key generada con ssh-keygen -t rsa, pero lo configure de tal manera para que terraform generara esa key automáticamente.

Puesto que no terminaba de conseguirlo volví a dejar que me obtuviera la public key desde ssh-keygen -t rsa.

1. Se deberá indicar la licencia utilizada e indicar las restricciones y el uso que permite la licencia.

La licencia utilizada para la práctica es Azure for Students donde cada alumno se registra con la cuenta de correo electrónico de la institución educativa. Esta licencia nos ofrece todos servicios de forma gratuita hasta agotar el crédito de 100 euros por un periodo de 12 meses.

Ejemplo de servicios disponibles:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente