Automatización de procesos en Azure a través de Terraform y Ansible

Autor: Javier Guillén González

Año: 2023

Notas del Autor:

- https://github.com/javk13/AzurePractica.git (master)

- https://github.com/javk13/ServidorWeb.git

- https://github.com/javk13/WebAks.git

Resumen

En este trabajo se explora la automatización de la comunicación entre el servicio Cloud Azure con las herramientas Terraform y Ansible.

El objetivo de la práctica es solicitar al proveedor Azure un repositorio de imágenes, una máquina virtual y un cluster de Kubernetes gestionado.

Para ello utilizaremos en primera instancia la herramienta Terraform, la cuál solicitará de forma automática la infraestructura al proveedor con una serie de características que de detallarán más a delante.

Posteriormente, a través Ansible realizaremos la configuración de dos servicios web.
Estos servicios estarán contenerizados y ambos harán uso de las imágenes de un repositorio proporcionado por Azure.

El primer servicio será un índice de Apache, que estará hospedado en una máquina virtual y será accesible desde internet en el puerto 8080 y tendrá un certificado x509 auto firmado.

El segundo servicio será un índice de Nginx hospedado en un nodo gestionado de Kubernetes, el cuál será accesible desde internet en el puerto 80 y hará uso de un volumen de almacenamiento persistente.

En esta práctica no se explora la automatización entre las herramientas Terraform – Ansible.

Método

Durante este informe se irá describiendo los pasos más destacados para mí y la aparición de errores. Describiéndolos, explicando las pruebas realizadas.
El propio código describe los pasos.

Esta práctica se ha realizado con éxito utilizando un sistema operativo host **Fedora 38**.
Para el aprovisionamiento de infraestructura con Terraform deberemos tener un par de **claves ssh que serán sustituidas** cada vez que ejecutemos las plantillas.
Se ha utilizado **Ansible 2.14.7** para automatización de la configuración con éxito y se han instalado los siguientes paquetes:

Git:
- sudo yum install git

Azure-Cli:
- sudo dnf install azure-cli
Azure-Imagen (Aceptación de terminos)
*- az vm image terms accept --urn almalinux:almalinux:9-gen2:9.1.2022122801

Terraform:
- sudo dnf install -y dnf-plugins-core
- sudo dnf config-manager --add-repo https://rpm.releases.hashicorp.com/fedora/hashicorp.repo
- sudo dnf install terraform

Ansible:
sudo dnf install ansible

- Colecciones:
  - ansible-galaxy collection install community.general
  - ansible-galaxy collection install community.crypto
  - ansible-galaxy collection install containers.podman
  - ansible-galaxy collection install kubernetes.core

- Programas:
  - sudo dnf install python3
  - sudo dnf install python3-dnf
  - sudo dnf install python3-passlib
  - sudo dnf install podman
  - sudo dnf install python3-kubernetes
  - sudo dnf install python3-PyYAML
- sudo dnf install python3-jsonpatch

Tras el proceso de Terraform deberemos incluir manualmente la IP de la máquina virtual en el documento host de Ansible y la contraseña del repositorio de imágenes en el playbook 00_playbook.yaml

Automatización de procesos en Azure a través de Terraform y Ansible

Comenzamos la práctica teniendo en cuenta que el autor nunca ha trabajado de informático, ni ha utilizado sistemas linux basados en rpm’s y comenzó a utilizar la Terminal en Febrero de 2023.

Infraestructura con Terraform:

VM:

Comienzo enumerando los elementos que necesito para la infraestructura.
Dado que no conozco en detalle todos los elementos que necesito, pero sí conozco el mapa general y en clase se ha realizado el aprovisionamiento de todos los recursos, aunque no de forma conjunta y tengo código en el repositorio, comienzo a crear código haciendo vm.

Problema: clave ssh.

El primer problema que me encuentro, que ya lo tuve cuando hice la primera práctica, es la clave ssh.
En su momento quise crear directamente las claves desde Terraform, y lo hice sin problema, añadiendo un recurso de 'tls_private_key'y anulando la variable de public key para entregarla directamente del terraform.tfstate

- Tras buscar diversa documentación en varias páginas,buscar otros métodos y realizar diversas pruebas, opto por copiar automáticamente la clave generada a mi ordenador a través del 'provisioner "local-exec"', tanto la privada como la pública, pero Terraform no me permite el uso de variable con un archivo no existente.
Esta variable es un requisito del caso práctico, si debo cumplirlo crearé manualmente un par de claves y dejaré el código con #, ya que me parece interesante desde la seguridad generar claves nuevas cada vez.

- Viendo los diferentes códigos del repositorio para la creación de diversos recursos, voy creando mi código a mi gusto. Aunque no tengo experiencia profesional me gusta el apoyarse mucho en las variables, de esta forma podemos reutilizar y adaptar el código con facilidad a través del archivo de variables.

- Consigo copiar ambas claves ssh a la cpu. El problema estaba en que al ser la variable 'public_key_path', debía tener el archivo creado anteriormente. Tras revisar el código del balanceador de carga utilizo la variable 'public_key' con exito. (Posteriormente, prácticamente al final de la práctica, me doy cuenta que no es así. Necesitaba tener un par de claves creadas)

Cluster Kubernetes:

Tras leer bastante documentación de Terraform en Azure, la página oficial de Hashicorp y en su repositorio de Github tengo bastantes dudas sobre cómo funciona el tema de las redes y poder asignar una ip pública diferente al nodo.

- Revisando grabaciones veo que la conexión con el cluster debe hacerse a través del API.

- He tenido algún problema con el dimensionamiento, ya que me pide un mínimo de 2 cpu's. También me está creando un segundo grupo de recursos, que tras revisar documentación es normal, pero en la clase no veo otro grupo de recursos al profesor. (Esta duda sigo sin tenerla resuelta a día de hoy)

Resto de recursos Terraform:

- Creación del resto de recursos con Terraform sin muchos problemas.

- Realizo conexión manual con vm, registry (descargando y subiendo imágenes de forma manual) y con el cluster para comprobación, siendo correcto.

Configuración con Ansible:

Destacar que inicialmente que instalado Centos 8 en VM y en mi localhost corre un sistema basado en Ubuntu 20.04.

Visualización de las grabaciones de clases para refrescar conceptos.
Aunque la estructura a través de roles parece muy interesante, opto por un sistema simple para no complicar el desarrollo de la práctica. Más a delante quizás pueda comprobar pasos comunes y reorganizar el sistema de playbooks.

Problema: Reconocimiento de módulos.

- Comienzo a definir tareas y encuentro problemas en el reconocimiento de los módulos relacionados con la creación del certificado. Tras revisar bien la documentación instalo las colecciones necesarias de la comunidad.

- Encuentro problema ya que el código de mi versión no soporta el módulo.
Reviso más a fondo la documentación y observo que tengo python 3.8 instalado, actualizando a 3.9 sin mejores resultados.

- Sigo mirando documentación, conectándome a la máquina, instalando manualmente el pass-lib a través de la activación del repo epel y haciendo comprobaciones del correcto funcionamiento del pass-lib.
Sigo y sigo y me doy cuenta que la documentación en Ansible relativa al módulo community.general.htpasswd está en la última versión.
Compruebo mi versión de Ansible, siendo la 2.9
Despliego documentación de versión 2.9 en web Ansible y veo el módulo correcto a utilizar.

- Añado tarea de activación de repositorio EPEL=8 en playbook y python3-passlib en variables, automatizando correctamente.(Lo que me ha costado…)

Problema: Configuración de aplicación web sobre Vm.

- Para poder automatizar el proceso, creo un repositorio con los archivos de configuración necesarios y añado módulo de descarga.

- Ansible no me permite la descarga en directorio con archivos existentes, así que creo uno nuevo y realizo copia.

- Me peleo bastante con el módulo de copia para determinar recurso remoto... y no consigo la copia de varios archivos a través de una variable. Opto por copia de carpeta.

- Me encuentro problemas de permisos y cambio a nivel general owner y tipo de permiso, funcionando correctamente.

- Confirmo manualmente la creación de imagen,systemd etc. y comienzo con módulos de Podman.

Problema: Versión Ansible.

- En la versión 2.9 de Ansible no tengo la funcionalidad de construir imagen desde archivo... podría instalar Docker que sí parece tenerlo.... Actualizo Ansible a 2.15

- Modifico módulos existentes (Por suerte había comentado algunos de ellos en vez de borrarlos)

Problema: Creación de imagen.

- Me peleo con atributos de creación de imagen desde archivo remoto y finalmente en el mismo módulo creo la imagen, la etiqueto y la envío al acr.

Problema: Acr

- En primer momento me autentifico con usuario y contraseña, posteriormente, al usar el módulo de creación de contenedor tengo algunos problemas con identificación en registry y opto por crear archivo .json para ambos módulos.

Servicio web Apache en Vm:

- Sorprendentemente consigo el resto de hitos sin mucho problemas.

-Disfruto de mi index.html configurado de forma automática

Ansible y Azure para Kubernetes:

- Veo videos (Otra vez) y realizo los ejercicios (Otra vez) para refrescar.

- Cuando comienzo con la elección de imagen me doy cuenta de que no conozco con profundidad el funcionamiento de cómo se comportan las aplicaciones y se complementan con otras para funcionar en front y back

- Opto por utilizar ejemplo de nginx ya que se encuentra como ejemplo en numerosos sitios y conozco ruta para almacenamiento persistente según práctica de clase.

- También veo que en los ejemplos de clase se nos muestra por separado deployments, services,persistent volume claims y pods.

Quizás los ejemplos de clase están pensados para poder hacer todo individualmente pero que tengamos que pensar un poco en cómo unir las piezas.

- Compruebo manualmente el proceso, y es correcto.
En la ruta no aparece ningún index.html y aparece error 403 de nginx al consultar el servicio, pero estoy accediendo a un servicio desde internet que hace uso de un almacenamiento persistente (Si puedo lo cambio pero estoy con poco tiempo y esto cumple con los requerimientos de la práctica)

- Defino mi .yaml y estructuro mentalmente el proceso. Repetir proceso de descarga de .yaml, imagen al registry, credenciales cluster y ejecución de .yaml remoto utilizando imagen del repositorio.

Observo que en la construcción desde Containerfile puedo nombrar, etiquetar y enviar la imagen a mi gusto, mientras que al usar una 'de serie' debo descargar, cambiar nombre + etiqueta y enviar.
(Supongo que podría crear un Containerfile en el Repositorio del yaml simplemente con FROM y realizar todo el proceso apoyándome solo en un módulo. Lo dejo así para probar diferentes formas)

Problema: Borrado de imágenes

Creo módulo de borrado, pudiendo realizarlo al nombrar una imagen individual pero dando fallo al introducir varias a través de variables. No me atasco ya que no es requisito, continuo y ya investigaré al terminar la practica.

Problema: Comando ‘az’

- El módulo de credenciales me da fallo en reconocimiento de comando 'az'. Tras ver video de clase me doy cuenta que estoy trabajando con host remoto y no con local.

- Creo un playbook con localhost para comprobar comando y no aparece fallo en reconocimiento de comando.

Ansible y Kubernetes:

Problema: Python3-Kubernetes (En realidad del Host)

- Aquí he tenido muchos problemas y no comprendí el proceso hasta muy avanzado.
Mientras paso las notas tomadas a este informe veo que realicé algunas pruebas que no tienen sentido.

- Al ejecutar el modulo de kubernetes aparece fallo de python3-kubernetes.
Compruebo versión, siendo la última versión disponible la 11 y necesito 12 o superior.

(El host era la Vm y no comprendí que los requisitos eran para el ordenador y host desde el que se ejecuta Ansible)

- Achaco el fallo al Centos 8 y su versión 11 de python3-kubernetes.

- Reviso imágenes disponibles y condiciones de facturación por uso, utilizando AlmaLinux 9 en un formato gratuito.

- Modifico código y ejecuto, teniendo fallo en kube-config file, no encontrando credenciales.

- Reviso la versión de python3-kubernetes en mí ordenador, siendo 7.0 y no teniendo soporte para llegar al 12. (Aquí empiezo a darme una ligera cuenta de lo que pasa)

Problema: Huida hacia adelante (Estas pruebas se realizan por intentar evitar cambiar el sistema operativo)

- Pruebo a instalar kubectl, colecciones y dependencias de Ansible y cli de Azure en vm.

- Aparecen fallos en Kube-config file relacionados con comandos apt no encontrados y solicitudes de login aún habiendo entrado manualmente en la vm y loguearme desde ahí.

Terraform y Fedora 38

- Instalo en otra unidad sistema operativo moderno, limpio y basado en rpm’s

- Compruebo documentación de Terraform y hay compatibilidad con versiones más antiguas, pero no aparece la 38.

Problema: Clave ssh Terraform

- No hay claves existentes y habría que vincular en admin_ssh_key, public_key con la nueva creación del archivo.
Aquí me doy cuenta que el recurso de copia modifica, no crea de nuevo.
(En alguna parte la documentación que he visto durante la práctica, he visto cómo decirle a Ansible el nombre del archivo de debe buscar para conectar por ssh, y me resulta interesante crear claves nuevas con un nombre diferente al id_rsa, utilizarlas y renovarlas cada vez que reconstruyamos)
Como la creación de clases no está contemplada en la práctica, copio manualmente dos claves existentes “ Que tienen correlación” y ejecuto de nuevo.

Problema: Caos y destrucción

Al ejecutar Terraform aparece fallo del provider con Acr.. remitiéndome a ponerme en contacto.

- Compruebo portal Azure desde cpu y me aparecen unos recursos, mientras que el mismo portal desde portátil tiene otros…
- Destruyo todo, cierro uno de los portales y ejecuto correctamente.
Fallo puntual de Azure.

Ansible y Fedora 38 “La Salvadora”

- Compruebo e instalo requisitos y colecciones según documentación Ansible.

- Ejecuto playbooks 'No Kubernetes' con éxito.

- Vuelvo a ver las grabaciones de Kubernetes y aclaro conceptos.

Problema: Recurso yaml.

- En mi primera definición del src apunté a un repositorio, sin éxito.(Modifico a ruta local y lo encuentra)

- Me fuerza a describir namespace aún siendo default.

Y…….

EJECUTO CON ÉXITO.

Para comprobar este éxito he tenido que conectarme con el Api para solicitar la Ip del servicio definido como load balancer, y en principio no he podido, ya que no tenía instalado el kubectl. Me ha parecido curioso que Ansible tenga otra manera de comunicarse con el Api.

Conclusiones

Azure y Terraform:

Durante el desarrollo de esta práctica he aprendido y ampliado mis conocimientos base sobre las infraestructuras.
Al tener que definir los recursos que necesito aprovisionar con Terraform, he ido aprendiendo la función de cada uno de ellos, las principales características que debo definir y cómo interactúan entre ellos.
Este apartado no me ha resultado especialmente difícil, he podido entender con facilidad la documentación de Hashicorp y al usar esta herramienta con Azure, he encontrado un sin fin de información y ejemplos en la plataforma de 'Microsoft Learn'.
Realmente he quedado impresionado con Microsoft.
Me quedo con un par de incógnitas a descubrir con Terraform, la creación de claves ssh en archivo específicos y el utilizar esta herramienta con un provider 'Local'.
Esto último me resulta interesante ya que todavía no tengo interconectados mis dispositivos particulares, pero poder crear automáticamente máquinas virtuales, compartir ciertos recursos y redes, cargar diferentes sistemas operativos he integrarlo con Ansible, me parece que tiene mucho potencial. La velocidad de creación, al menos con Azure, es muy rápida, y podría aprovisionar recursos 'al vuelo' cuando realmente tenga que hacer uso de ellos.
A nivel profesional me parece maravilloso. He podido comentar con amigos de un sector informático algo más 'tradicional' estas herramientas, y los tiempos que escucho en cuanto a aprovisionamiento de infraestructuras, su control y adaptabilidad no tienen nada que ver con Terraform (Desde la ignorancia pienso 'Si esto que te lleva varias horas, una vez creadas las plantillas de Terraform modificas el archivo de variables y corre solo'. Pero yo no trabajo profesionalmente de ello.) Y esta es otra razón más para preguntarme a cerca del uso de Terraform 'On-premise'

Ansible:

Si Terraform es el amigo simpático que te facilita todo, Ansible es el amigo 'toca huevos' que todo te lo pone en duda. Tiene un buen trasfondo, en realidad es un tipo muy majo y salen grandes cosas de esa amistad, pero madre mía lo que hay que pelear.
La primera lección que obtengo es el nivel de detalle al consultar la información.
Tiene una documentación muy completa que te explica en detalle todos los atributos, opciones etc que tiene, pero ha que leerlos con detenimiento hasta entenderlos e interiorizarlos.
Señalar también la compatibilidad de versiones. Básico para trabajar con los módulos y he tenido que pelear con ello hasta decir basta. Realmente nunca había experimentado nada parecido.
En cuanto a la propia herramienta ' en si' un cañón.
La herramienta es muy potente, para que tiene una integración enorme con todo y se pueden hacer maravillas. Eso sí, hay que tener muy claros los procesos manuales, 'lo que se hace detrás' y el funcionamiento con sus particularidades de cada módulo.

Cuando he comenzado a moverme con algo de soltura en la documentación y he probado diversas estrategias en diferentes módulos, he visto que se puede hacer lo que quieras mientras tú te adaptes a la herramienta.
Como tuve que reiniciar la práctica en un sistema operativo nuevo, ya tengo los pasos manuales que dí para crearme mis propias plantillas para mis dispositivos.
A nivel de usuario me parece una herramienta muy útil que permite conservar y ampliar la configuración personal en cualquier dispositivo.
Y esta idea solo viendo los pocos módulos propuestos por el profesor y alguno más que he buscado para algún paso muy particular.
La herramienta con la que más he sufrido con diferencia y la que creo que más utilizaré.
Me quedo con la espina del uso de roles.

Y debo mencionar también a Kubernetes, un mundo en sí mismo que hay que comprender muy bien para poder interactuar y sobre todo automatizar sus procesos.

Me hubiese gustado (Y tengo algo de documentación guardada al respecto) disponer de más tiempo para automatizar las salidas de Terraform con Ansible y su ejecución automática.
Y con más tiempo la mente empieza a volar, a pensar el varias vm’s con un balanceador delante y ubicadas en otra localización para evitar las restricciones.
Una máquina central sobre la que cargar todo y que funcione como ejecutora de Terraform, Nodo de control de Ansible y desde donde se ejecuten las órdenes a Kubernetes (Y obtenga credenciales válidas).

He sufrido con la practica, la he dedicado una barbaridad de horas, me he roto la cabeza en muchas ocasiones (Cada paso con Ansible una nueva desgracia) y en algunos momentos me he desesperado y casi bloqueado, pero comparo lo que sabía hace tres semanas respecto ahora y no doy crédito.

Muchas gracias por el desafío.

Javier Guillén González