

ALUMNO: HERNÁN JAVIER CASTEL

INDICE CONTENIDO

INTRODUCCION.....	2
OBJETIVO.....	4
DISEÑO DE LA SOLUCION	5
DESPLIEGUE INFRAESTRUCTURA EN AZURE.....	6
DESPLIEGUE EN AZURE	7
DIAGRAMA	9
PROBLEMAS ENCONTRADOS	10

INTRODUCCION

En el marco del curso de DevOps & Cloud para el caso práctico número dos, se provisionará la infraestructura en Azure , la cual da soporte a las siguientes requerimientos.

- Desplegar un web server contenerizado en una máquina virtual.
- Instalar una aplicación en el clúster de Kubernetes.

Para desplegar la infraestructura como código se utilizará Terraform que a partir del código sin necesidad de realizar configuraciones manuales armará el ambiente en la infraestructura de la Cloud de Azure.

Utilizamos ansible que una aplicación OpenSource para gestionar la configuración de las aplicaciones requeridas de forma totalmente automatizada.

En Terraform, se empleó un archivo de credenciales para cargar los datos necesarios de acceso a la cuenta de Azure. Este archivo contiene la información de autenticación requerida para establecer la conexión con los servicios de Azure y realizar las operaciones pertinentes en la infraestructura de la nube.

Para el despliegue de la aplicación, se adoptó una convención de nomenclatura donde cada componente se identifica como "jenkins_componente". Esta práctica se implementó con el propósito de mejorar la comprensión del funcionamiento de Kubernetes, evitando ambigüedades al referirse a los distintos elementos.

Se procuró emplear librerías de Ansible en la medida de lo posible para ejecutar las tareas necesarias. En casos donde no fue factible encontrar una librería adecuada, se recurrió al uso del módulo "command" para ejecutar comandos directamente en los sistemas afectados

OBJETIVO

Desplegar una aplicación con un enfoque completamente automatizado dentro de una máquina virtual utilizando Podman, en donde la imagen que se utiliza esta previamente subida en el Container Registry de Azure. La imagen por utilizar es nginx

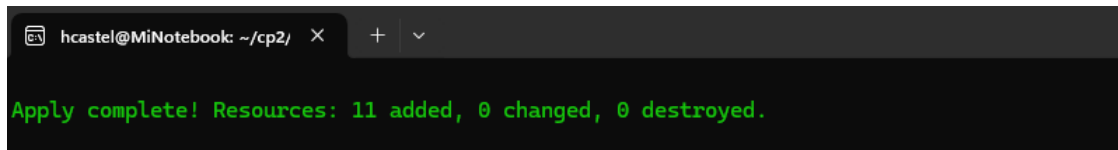
Instalar una aplicación en el Cluster de Kubernetes autogestionado por Azure, aprovechando las características de SaaS disponibilizada por Azure.

La aplicación previamente subida al Container Registry de Azure es Jenkins.

DISEÑO DE LA SOLUCION

Infraestructura en Terraform

En la infraestructura desplegada con Terraform, se crearon en total 11 recursos en Azure

A terminal window with a dark background. The title bar shows 'hcastel@MiNotebook: ~/cp2/'. The terminal output is green text on a black background: 'Apply complete! Resources: 11 added, 0 changed, 0 destroyed.'

Para la VM se utilizó el sistema operativo CentOS 8 versión 22.03.28 y la vm del tipo Standard_D1_v2

Se utilizo la locación East US, ya que por motivos de ubicación (estoy en Argentina) me pareció más apropiada por un tema de latencia si supongo que los usuarios se conectan desde Latinoamérica.

La Estructura de directorio es la siguiente con su respectiva descripción

```
cp2
|--
|-- Terraform
|   |-- credentials.tf           [Variables con las credenciales de conexión a Azure]
|   |-- variables.tf             [variables comunes del aprovisionamiento]
|   |-- main.tf                  [Creación de recursos principales de AZ]
|   |-- resource_acr.tf          [Creación ACR]
|   |-- resource_aks.tf          [Creación del AKS]
|   |-- resource_network.tf      [Creación de la virtual network, subnet, nic, ip]
|   |-- resource_security.tf     [Creación network security, reglas]
|   |-- resource_vm.tf           [Creación Virtual machine]
|   |-- outpust_acr.tf           [Variables de salida del acr]
|   |-- output_vm.tf            [Variables de salida de la Virtual machine]
|
|-- deploy_azure.sh              [Script para desplegar la infraestructura]
|-- destroy_azure.sh            [Script para destruir la infraestructura]
```

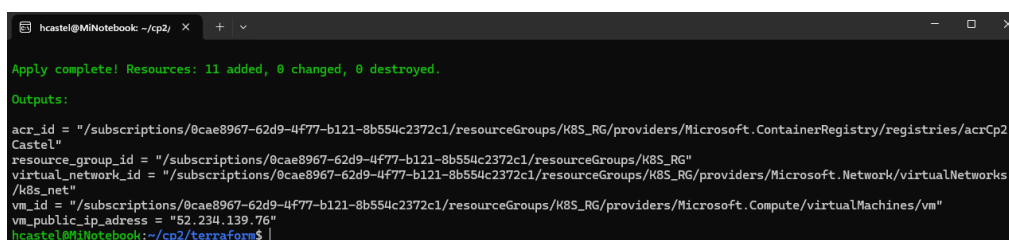
DESPLIEGUE INFRAESTRUCTURA EN AZURE

Para implementar la infraestructura en Azure, es necesario seguir un conjunto de pasos específicos:

- Clonar el repositorio <https://github.com/hernancastel/cp2.git> en la maquina local donde se ejecutará.
- Se tiene que validar que exista la llave publica, se puede ejecutar el comando `cat ~/.ssh/id_rsa.pub`, en caso que no exista hay que crearla por ejemplo con el comando `ssh-keygen -t rsa -b 4096 -C "tu_email@example.com"`
- Completar las siguientes variables del archivo que se encuentra en la carpeta *terraform/credentials.tf* donde se clono el repositorio, con los valores de su suscripción de Azure:
 - subscription_id
 - client_id
 - client_secret
 - tenant_id
- Para crear la infraestructura. ejecutar el archivo *deploy_azure.sh*

Luego de finalizar la ejecución se visualizarán los outputs que se utilizan en el ansible, resguardar los valores:

- acr_id
- resource_group_id
- virtual_network_id
- vm_id
- vm_public_ip_adress



```
hcastel@MiNotebook: ~/cp2/ x + - □ x
Apply complete! Resources: 11 added, 0 changed, 0 destroyed.
Outputs:
acr_id = "/subscriptions/0cae8967-62d9-4f77-b121-8b554c2372c1/resourceGroups/K8S_RG/providers/Microsoft.ContainerRegistry/registries/acrCp2Castel"
resource_group_id = "/subscriptions/0cae8967-62d9-4f77-b121-8b554c2372c1/resourceGroups/K8S_RG"
virtual_network_id = "/subscriptions/0cae8967-62d9-4f77-b121-8b554c2372c1/resourceGroups/K8S_RG/providers/Microsoft.Network/virtualNetworks/K8s_net"
vm_id = "/subscriptions/0cae8967-62d9-4f77-b121-8b554c2372c1/resourceGroups/K8S_RG/providers/Microsoft.Compute/virtualMachines/vm"
vm_public_ip_adress = "52.234.139.76"
hcastel@MiNotebook:~/cp2/terraform$
```

DESPLIEGUE EN AZURE

Descripción la estructura de los archivos de Ansible.

```
|-- cp2
|   |-- terraform
|   |-- ansible
|       |-- install_podman.yaml [Instala Podman en vm]
|       |-- webserver_acr_playbook.yaml [Pull de nginx docker.io y la sube al ACR]
|       |-- webserver_container_playbook.yaml [imagen contenerizada en la VM]
|       |-- aks_playbook.yaml [Crea componentes del Cluster]
|       |-- aks_deploy_playbook.yaml [Deploy de la aplicación en Aks]
|       |-- all_playbook [Importación de los de playbook]
|       |-- container-nginx.service [Conf systemd para servicio web server]
|
|-- deploy_ansible.sh [Script para desplegar la solución]
```

Como ya tenemos descargado el repositorio <https://github.com/hernancastel/cp2.git>

Porque tuvimos que correr terraform, lo primero que tenemos que hacer es cargar las variables que utilizaremos para desplegar, el archivo con las variables está ubicado en:

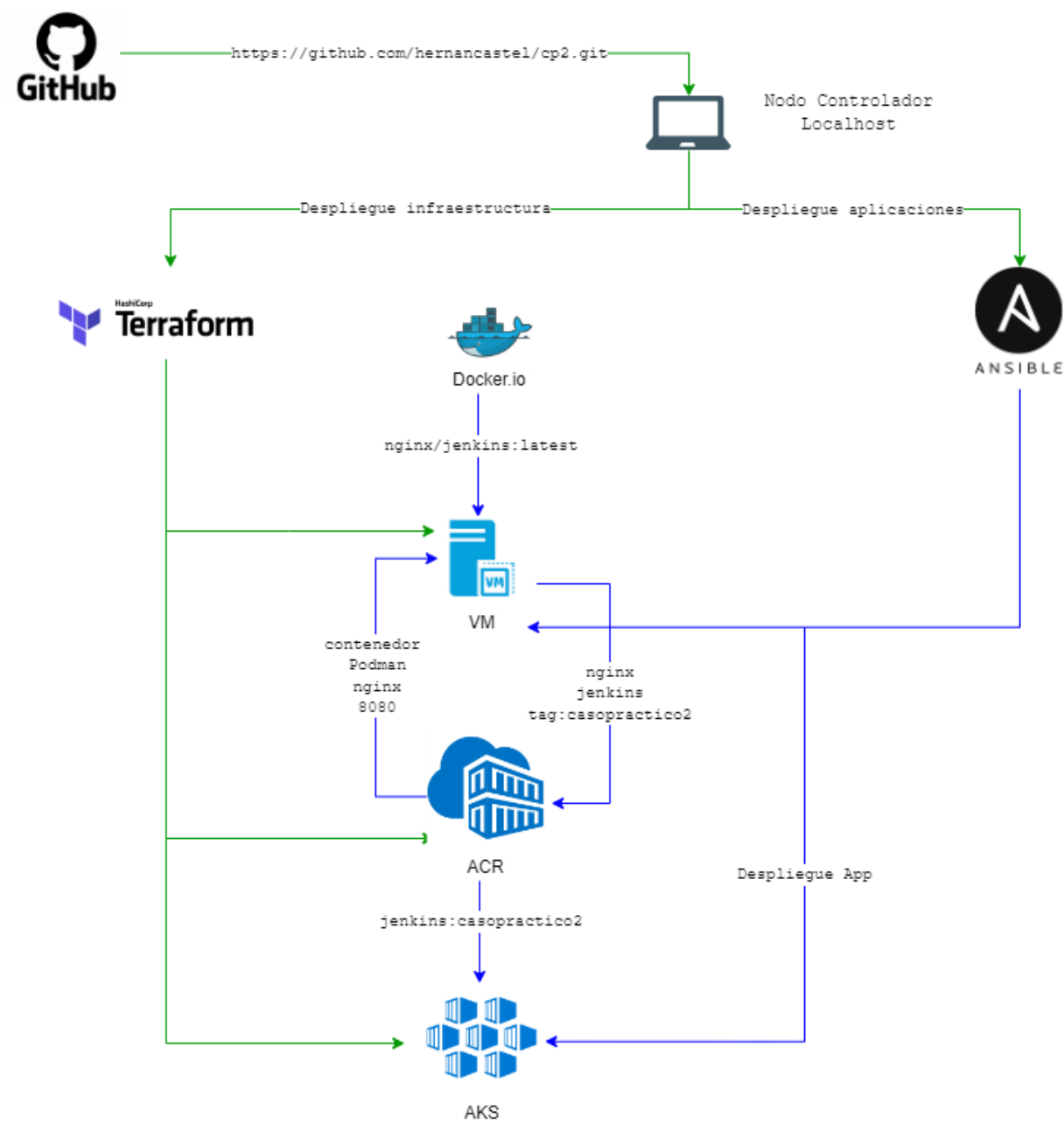
[/ansible/vars/vars_global.yaml](#) e ingresar los valores correspondientes:

acr_username: output de terraform `vm_public_ip_adress`
acr_password: lo obtenemos de la cuenta de AZ o con el cliente AZ
(Ingresar al portal de AZ e ir a Container Registry > claves de acceso)
acr_server: nombre que definimos en terraform + `azurecr.io`
az_username: Login en Azure
az_pass: password de Azure
resource_group: Nombre del rg que definimos en terraform
aks_name: Nombre del aks que definimos en terraform
tenant_id: tenant_id de la cuenta de AZ
client_id: El mismo que utilizamos para desplegar Terraform
client_secret: El mismo que utilizamos para desplegar Terraform

Una vez configuradas las credencias ejecutar el sh en la carpeta principal del repositorio

Ejecutar el archivo [*deploy ansible.sh*](#) para desplegar la aplicación web server en nuestro caso nginx contenerizada con Podman y también deploya la aplicación Jenkins en el clúster de Aks, además de subir al Acr las imágenes.

DIAGRAMA



PROBLEMAS ENCONTRADOS

Sistemas Operativo Linux. La realización de este proyecto me brindó la oportunidad de adentrarme en el mundo de Linux y profundizar en su funcionamiento, a pesar de no ser un usuario de este sistema operativo. Este desafío me permitió explorar y comprender aspectos fundamentales de Linux.

No era posible acceder a la vm desplegada por Azure por ssh. Investigando puede ver el problema era con clave publica `id_rsa.pub` modificando el despliegue y generando la clave pública logre registrarla de forma automática y poder conectarme.

Dependencias de librerías necesarias para ejecutar Ansible, me llevo tiempo terminar y ver que módulos necesita Ansible para que corra correctamente los playbooks. Por ejemplo Python , ya que Ansible está escrito en Python, por lo que se necesita tener Python instalado en tu sistema. Se recomienda utilizar Python 3.x, ya que algunas versiones más recientes de Ansible pueden no ser compatibles completamente con Python 2.x.

OpenSSH: Ansible utiliza ssh para conectarse a los servidores remotos y ejecutar comandos en ellos. Después de probar varias veces, me di cuenta de que no tenía instalado el ssh, lo instalé y puede avanzar.

En ansible quería utilizar módulos que al instalarlos después no me los reconoce cuando lanzaba los playbook, también creo que se debe a que estaba usando el sistema WSL de Windows, que es una característica de Windows que permite ejecutar un entorno de línea de comandos de Linux directamente en Windows. WSL permite a los usuarios ejecutar distribuciones de Linux, como Ubuntu, Debian, Fedora, etc., de forma nativa en Windows, sin necesidad de usar máquinas virtuales o emuladores.

A modo de ejemplo, instalaba alguno de los módulos y después cuando los corría no los reconocía. *ansible-galaxy collection install community.docker*
luego varios reinicios de Windows y cerrar y abrir la terminal de Linux, puede solucionar con los módulos que utilice para el caso práctico.

Se podría mejorar los temas de claves, secretos y datos sensibles, para este trabajo practico asumí que no hay riesgo, quedaría pendiente mejorar este punto.

En el clúster de Kubernetes, habilite la comunicación entrante para llegar a la aplicación desplegada, por un tema tiempos de investigación debería mejorarse ese punto para restringir el acceso de posibles ataques.