Experto Universitario en DevOps & Cloud

CASO PRACTICO N° 2

Autor: José Alejandro Dobla

Proyecto: Caso Práctico N° 2

INFORMACIÓN GENERAL

El propósito de este documento es para dar una descripción de la implementación realizada para el Caso Práctico N° 2.

En dicho caso práctico se solicitó realizar el despliegue de infraestructura y su respectiva configuración de modo automatizado en el proveedor Cloud Azure.

Para dicha automatización se requirió la utilización de herramientas tales como Terraform y Ansible y Helm.

La práctica realizada se llevó a cabo a nivel local con pruebas utilizando Vagrant como así también hemos desplegado la infraestructura en el proveedor Cloud. Dado que hemos llevado a cabo mayor trabajo con la parte Cloud, es este despliegue el que estaremos detallando y no asi el local con Vagrant.

Adicionalmente como parte del requerimiento se solicitó adjuntar una documentación correspondiente donde se explique con mayor detalle posible toda la implementación. A tales fines, hemos preparado este documento con conjunto con el README que se encuentra en el repositorio de código el cual indicaremos en el documento.

AUDIENCIA

La finalidad del documento es brindar un detalle técnico y conceptual de lo realizado por lo que este documento está destinado a personal técnico, profesores como así también directivos.

PRE REQUISITOS

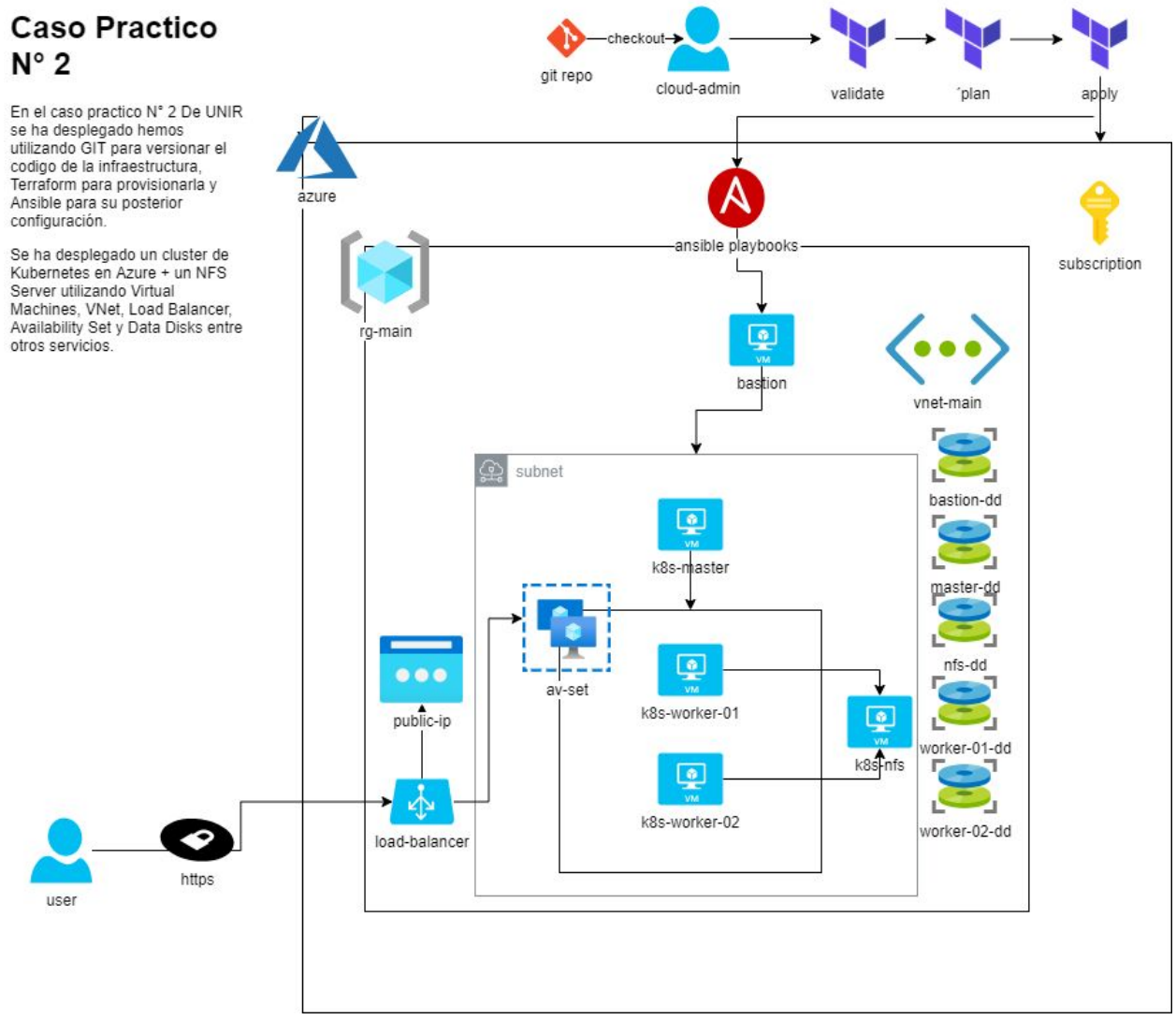
Para poder desplegar esta infraestructura hemos utilizado las siguientes herramientas y cuentas, las cuales serán necesarias para su correcto despliegue.

- Terraform
- Ansible
- AZ Cli
- Suscripción de Azure
- Powershell / Bash

DISEÑO DE INFRAESTRUCTURA

A continuación mostraremos un diagrama de la infraestructura que se desplegó.

Para el diseño de la infraestructura como código hemos desarrollado módulos locales de Terraform, Roles y Templates con Ansible.



Como se puede observar en el diseño, hemos utilizado servicios tales como Load Balancers, Availability Sets, Virtual Machines, Network Interfaces, Public IPs, Data Disks, Storage Accounts, Virtual Network y Subnets.

La provisión de la infraestructura se realizó con Terraform, la configuración de la misma como el despliegue de las aplicaciones se realizó con Ansible y Helm vía playbooks, utilizando el bastión.

DETALLE DE INFRAESTRUCTURA

A continuación un detalle de la infraestructura desplegada

VIRTUAL MACHINES

Role	Sistema Operativo	vCPUs	Memoria (GiB)	Disco Duro	IP Privada
master	CentOS 8.3	3.5	7	1 x 40 GiB (os), 1 x 10 GiB (data)	192.168.1.10/24
worker-01	CentOS 8.3	3.5	7	1 x 40 GiB (os), 1 x 10 GiB (data)	192.168.1.11/24
worker-02	CentOS 8.3	3.5	7	1 x 40 GiB (os), 1 x 10 GiB (data)	192.168.1.12/24
nfs	CentOS 8.3	3.5	7	1 x 40 GiB (os), 1 x 10 GiB (data)	192.168.1.13/24
bastion	CentOS 8.3	3.5	7	1 x 40 GiB (os), 1 x 10 GiB (data)	192.168.1.14/24

NETWORKING

Tipo	CIDR	IP Privada	IP Publica
VNET	192.168.1.0/24	-	-
Subnet	192.168.1.0/24	-	-
Public IP (Bastion)	-	-	Disponible post Apply
Public IP (LB)	-	-	Disponible post Apply
Load Balancer	-	-	-

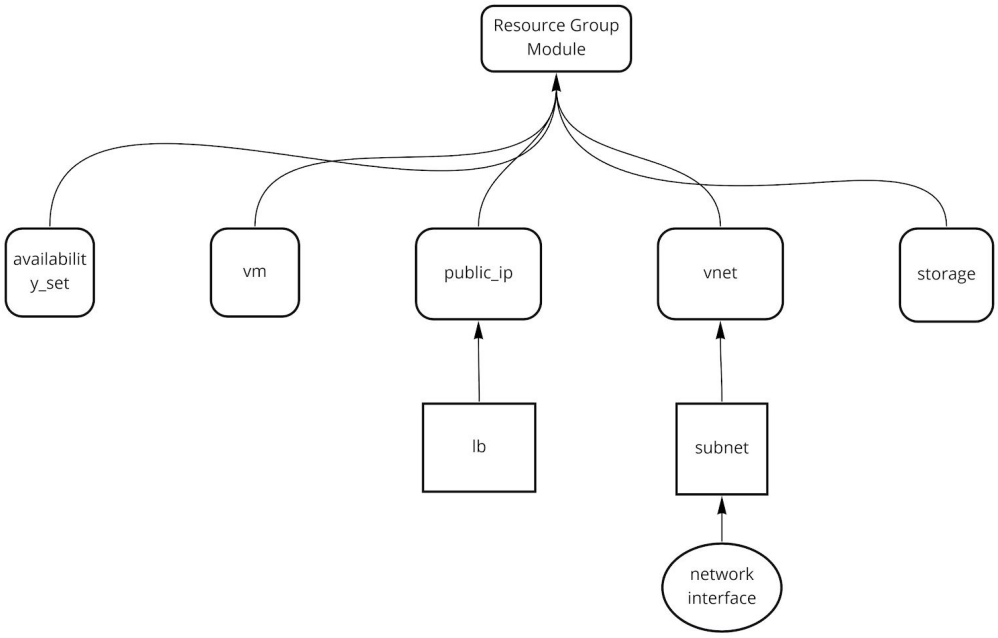
Network Interface (Master)	-	192.168.1.10	
Network Interface (Worker1)	-	192.168.1.11	
Network Interface (Worker2)	-	192.168.1.12	
Network Interface (NFS)	-	192.168.1.13	
Network Interface (Bastion)	-	192.168.1.14	-

STORAGE

Tipo	Tamaño	Recurso Asociado
Storage Account	n/a	Terraform TF State
Data Disk	30Gb	Master
Data Disk	30Gb	Worker1
Data Disk	30Gb	Worker2
Data Disk	30Gb	NFS
Data Disk	30Gb	Bastion

DEPENDENCIA DE MÓDULOS TERRAFORM

Hemos desarrollado un pequeño diagrama de la dependencia de módulos de terraform creados los cuales se muestran a continuación



DESPLIEGUE DE LA INFRAESTRUCTURA Y APLICACIÓN

Tanto el despliegue de la infraestructura como de la aplicación la hemos automatizado vía Terraform.
Para desplegar la infraestructura y su correspondiente configuración debemos realizar los siguientes pasos.

1.- Inicialmente clonamos el repositorio

```
$ git clone https://github.com/tareafina/UNIR_TP02
```

2.- Una vez instalada y configurada la Azure CLI ejecutamos el script de configuración. Agregar en el script de configuración la Subscription Key correspondiente, luego de la ejecución de este script obtendremos las key y storage account para el paso 3. Adicionalmente copiamos nuestra pub key para poder acceder a los servidores

```
chmod +rx ./scripts/azure_configuration_script.sh
```

```
./scripts/azure_configuration_script.sh

cp $HOME/.ssh/id_rsa.pub ./terraform/modules/vm/keys/id_rsa.pub

cp $HOME/.ssh/id_rsa ./terraform/modules/provision/keys/id_rsa
```

A.- Al ejecutar el script anterior, el mismo nos creará en nuestra suscripción de Azure los siguientes recursos:

- Storage Account para salvar el tfstate.
- Service Principal / App Registration utilizado para que Terraform pueda interactuar con ARM.
- Detalles de lo desplegado para su posterior utilización.

3.- En el archivo terraform/provider.tf completamos según corresponda

```
storage_account_name = ""

subscription_id = ""

client_id       = ""

client_secret   = ""

tenant_id       = ""
```

4.- Una vez configurado el provider y el backend procederemos a iniciar Terraform

```
terraform init
```

5.- Luego validamos la configuración

```
terraform validate
```

6.- Ejecutamos el plan

```
terraform plan
```

A.- En el plan/apply que hemos realizado desde nuestro local lo hemos subido al repositorio en la ubicación terraform/tfplan.txt

7.- Aplicamos los cambios y aguardar finalización

```
terraform apply
```

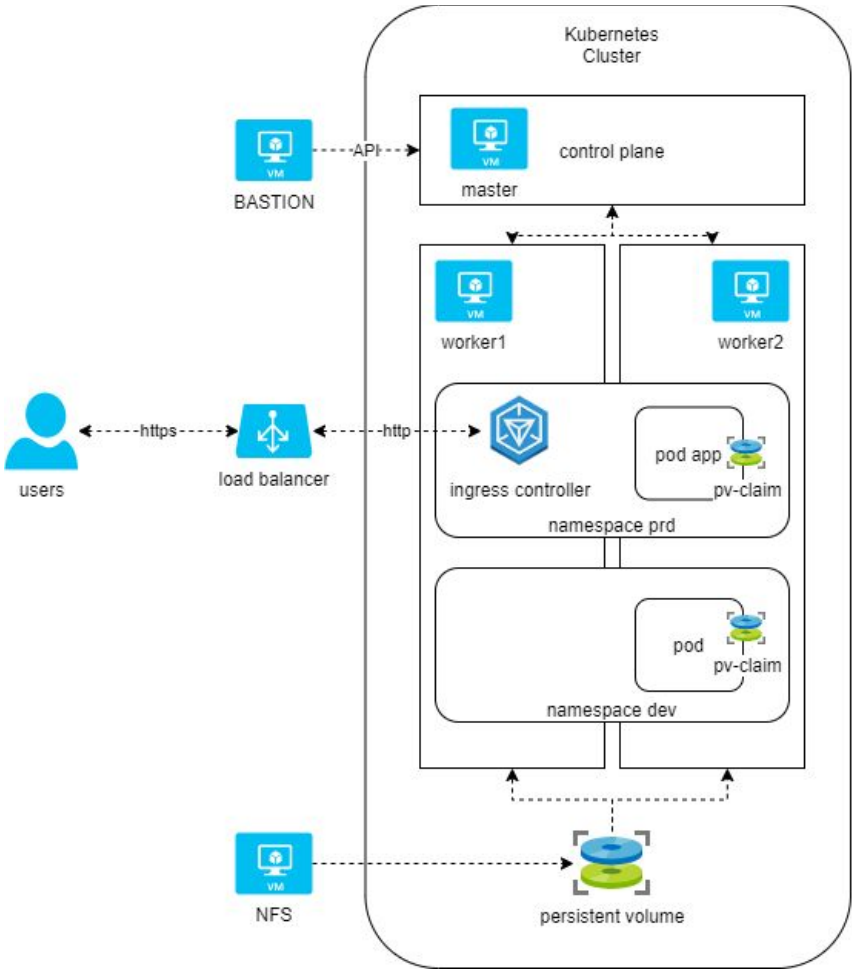
De este modo hemos desplegado y configurado vía playbooks de ansible la infraestructura para nuestra aplicación. Luego del Apply obtendremos la Public IP y FQDN Del Bastión y el Load Balancer via outputs.

DIAGRAMA DE KUBERNETES

Como parte de la implementación al provisionar el cluster de kubernetes se desplegaron las apps en los namespaces de “prd” (Prod) y “dev” (Develop) .

El Persistent Volume del NFS como sus respectivos Persistent Volume Claim para cada app fue uno de los desafíos que podrían realizarse.

A continuación se muestra un breve diagrama del Cluster y su futura definición.



ADICIÓN DE RECURSOS

En caso de precisar modificar alguno de los recursos desplegados o bien querer adicionar un nuevo recurso reutilizando los módulos de terraform, solo debemos gestionarlo vía el terraform/main.tf , archivo en el cual se encuentran declarados cada uno de los recursos a desplegar tomando ventaja de la reutilización de módulos.

OPORTUNIDADES DE MEJORA

Como parte de las oportunidades de mejora para este proyecto hemos detectado rápidamente las siguientes:

- Configuración del PV/PVC para las aplicaciones con el NFS
- Despliegue completo de servicios y configuración del Ingress Controller para las aplicaciones

- Pipeline de CI/CD para la infraestructura como sus aplicaciones

CONCLUSIÓN

A través del práctico realizado hemos podido automatizar gran parte de las tareas tales como la provisión y despliegue de la infraestructura y su respectiva configuración, también hemos podido desplegar unas aplicaciones de prueba en el cluster de Kubernetes.

Hemos identificado inicialmente algunas oportunidades de mejora para este caso práctico como así también hemos dado un valor añadido creando los módulos locales de Terraform para una posterior reutilización de los mismos.

El caso práctico ha sido un verdadero y real desafío que desde el momento uno ha valido la pena la inversión de tiempo e investigación para poder cumplir con el mismo.

Desde ya muy agradecido por la oportunidad, ha sido una excelente experiencia.