

Experto Universitario en Devops & Cloud

Automatización de despliegues en entornos Cloud

Curso 2020 - 2021

Antonio Ruiz Rondán

Cádiz, 17 de julio de 2021

ÍNDICE

1. Descripción del entorno	4
2. Procedimiento de despliegue	8
2.1. Despliegue de la infraestructura	9
2.2. Despliegue de kubernetes	11
3. Despliegue de la aplicación	15
3.1. Despliegue de la aplicación	15
3.2. Acceso a la aplicación	17
4. Problemas encontrados	18
5. Referencias	19

1. Descripción del entorno

Este informe corresponde al Caso Práctico 2 del Curso de *Experto Universitario en Devops & Cloud* de UNIR, en el cual se pide el despliegue de un clúster de Kubernetes en Azure utilizando Terraform y Ansible. No se puede para ello utilizar AKS.

En un principio se solicitaban que debían componerse de 4 máquinas virtuales, 1 master, 2 workers y un nodo que actúa de servidor NFS para almacenamiento, pero debido a las limitaciones de la cuenta Azure student de 4 vCPU y ya que la máquina que actúa como master necesita al menos 2 de ellos, hemos tenido que reducir a sólo 1 worker. Quedándonos así:

Role	Sistema Operativo	Tipo	vCPU	Memoria (GiB)	Disco Duro
Nfs	CentOS 8	Standard_DS1_v2	1	4	1 x 30 GiB
Master	CentOS 8	Standard_D2s_v3	2	8	1 x 30 GiB
Worker01	CentOS 8	Standard_DS1_v2	1	4	1 x 30 GiB

Vamos a desplegarlas en una subred 192.168.1.0/24 las IP privadas de las máquinas serán:

Nombre	IP
nfs.acme.es	192.168.1.115/24
master.acme.es	192.168.1.110/24
worker01.acme.es	192.168.1.111/24

En un principio se hizo plan de Terraform con las 4 máquinas solicitadas pero como puede verse, al intentar desplegarlo, éste era el mensaje que nos devolvía Azure:

```
....
....
azurerm_linux_virtual_machine.NFS: Creating...
azurerm_linux_virtual_machine.WORKER02: Creating...
azurerm_linux_virtual_machine.WORKER01: Creating...
azurerm_linux_virtual_machine.MASTER: Creating...
azurerm_linux_virtual_machine.NFS: Still creating... [10s elapsed]
azurerm_linux_virtual_machine.WORKER02: Still creating... [10s elapsed]
azurerm_linux_virtual_machine.WORKER01: Still creating... [10s elapsed]
azurerm_linux_virtual_machine.MASTER: Still creating... [10s elapsed]
azurerm_linux_virtual_machine.WORKER01: Still creating... [20s elapsed]
azurerm_linux_virtual_machine.MASTER: Still creating... [20s elapsed]
azurerm_linux_virtual_machine.WORKER01: Still creating... [30s elapsed]
azurerm_linux_virtual_machine.MASTER: Still creating... [30s elapsed]
azurerm_linux_virtual_machine.WORKER01: Still creating... [40s elapsed]
azurerm_linux_virtual_machine.MASTER: Still creating... [40s elapsed]
azurerm_linux_virtual_machine.WORKER01: Still creating... [50s elapsed]
azurerm_linux_virtual_machine.MASTER: Still creating... [50s elapsed]
azurerm_linux_virtual_machine.WORKER01: Still creating... [1m0s elapsed]
azurerm_linux_virtual_machine.MASTER: Still creating... [1m0s elapsed]
azurerm_linux_virtual_machine.WORKER01: Still creating... [1m10s elapsed]
azurerm_linux_virtual_machine.MASTER: Still creating... [1m10s elapsed]
azurerm_linux_virtual_machine.WORKER01: Still creating... [1m20s elapsed]
azurerm_linux_virtual_machine.MASTER: Still creating... [1m20s elapsed]
azurerm_linux_virtual_machine.WORKER01: Still creating... [1m30s elapsed]
azurerm_linux_virtual_machine.MASTER: Still creating... [1m30s elapsed]
azurerm_linux_virtual_machine.WORKER01: Still creating... [1m40s elapsed]
azurerm_linux_virtual_machine.MASTER: Still creating... [1m40s elapsed]
azurerm_linux_virtual_machine.WORKER01: Still creating... [1m50s elapsed]
```

```

azurerm_linux_virtual_machine.MASTER: Still creating... [1m50s elapsed]
azurerm_linux_virtual_machine.WORKER01: Creation complete after 1m51s [id=/subscriptions/6805d1d6-081c-49a6-ab0d-142248d48838/resourceGroups/kubernetes_rg/providers/Microsoft.Compute/virtualMachines/miWORKER01]
azurerm_linux_virtual_machine.MASTER: Still creating... [2m0s elapsed]
azurerm_linux_virtual_machine.MASTER: Still creating... [2m10s elapsed]
azurerm_linux_virtual_machine.MASTER: Still creating... [2m20s elapsed]
azurerm_linux_virtual_machine.MASTER: Still creating... [2m30s elapsed]
azurerm_linux_virtual_machine.MASTER: Still creating... [2m40s elapsed]
azurerm_linux_virtual_machine.MASTER: Still creating... [2m50s elapsed]
azurerm_linux_virtual_machine.MASTER: Creation complete after 2m50s [id=/subscriptions/6805d1d6-081c-49a6-ab0d-142248d48838/resourceGroups/kubernetes_rg/providers/Microsoft.Compute/virtualMachines/miMASTER]
|
| Error: creating Linux Virtual Machine "miNFS" (Resource Group "kubernetes_rg"):
| compute.VirtualMachinesClient#CreateOrUpdate: Failure sending request: StatusCode=0 -- Original
| Error: autorest/azure: Service returned an error. Status=<nil> Code="OperationNotAllowed"
| Message="Operation could not be completed as it results in exceeding approved standardDSv3Family
| Cores quota. Additional details - Deployment Model: Resource Manager, Location: westeurope, Current
| Limit: 4, Current Usage: 4, Additional Required: 2, (Minimum) New Limit Required: 6. Submit a
| request for Quota increase at
| https://aka.ms/ProdportalCRP/?#create/Microsoft.Support/Parameters/%7B%22subId%22:%226805d1d6-081c-49a6-ab0d-142248d48838%22,%22pesId%22:%2206bfd9d3-516b-d5c6-5802-169c800dec89%22,%22supportTopicId%22:%22e12e3d1d-7fa0-af33-c6d0-3c50df9658a3%22%7D by specifying
| parameters listed in the 'Details' section for deployment to succeed. Please read more about quota
| limits at https://docs.microsoft.com/en-us/azure/azure-supportability/per-vm-quota-requests."
|
|   with azurerm_linux_virtual_machine.NFS,
|   on vm.tf line 4, in resource "azurerm_linux_virtual_machine" "NFS":
|     4: resource "azurerm_linux_virtual_machine" "NFS" {
|
|
|
| Error: creating Linux Virtual Machine "miWORKER02" (Resource Group "kubernetes_rg"):
| compute.VirtualMachinesClient#CreateOrUpdate: Failure sending request: StatusCode=0 -- Original
| Error: autorest/azure: Service returned an error. Status=<nil> Code="OperationNotAllowed"
| Message="Operation could not be completed as it results in exceeding approved standardDSv3Family
| Cores quota. Additional details - Deployment Model: Resource Manager, Location: westeurope, Current
| Limit: 4, Current Usage: 4, Additional Required: 2, (Minimum) New Limit Required: 6. Submit a
| request for Quota increase at
| https://aka.ms/ProdportalCRP/?#create/Microsoft.Support/Parameters/%7B%22subId%22:%226805d1d6-081c-49a6-ab0d-142248d48838%22,%22pesId%22:%2206bfd9d3-516b-d5c6-5802-169c800dec89%22,%22supportTopicId%22:%22e12e3d1d-7fa0-af33-c6d0-3c50df9658a3%22%7D by specifying
| parameters listed in the 'Details' section for deployment to succeed. Please read more about quota
| limits at https://docs.microsoft.com/en-us/azure/azure-supportability/per-vm-quota-requests."
|
|   with azurerm_linux_virtual_machine.WORKER02,
|   on vm.tf line 136, in resource "azurerm_linux_virtual_machine" "WORKER02":
|  136: resource "azurerm_linux_virtual_machine" "WORKER02" {
|
|

```

Como decíamos, habíamos sobrepasado el límite de 4 vCPU y podíamos solicitar un aumento de cuota a Azure, lo cual también se intentó, pero con el siguiente resultado:

Nueva solicitud de soporte técnico ...

1. Descripción del problema

2. Solución recomendada

3. Detalles adicionales

4. Revisión y creación

Cuéntenos su problema y le ayudaremos a resolverlo.

Proporcione información acerca de la facturación, suscripción, administración de cuotas o problema técnico (incluidas las solicitudes de asesoramiento técnico).

Tipo de problema *

Límites de servicio y suscripción (cuotas) ▼

Suscripción *

Azure para estudiantes (6805d1d6-081c-49a6-ab0d-142248d48838) ▼

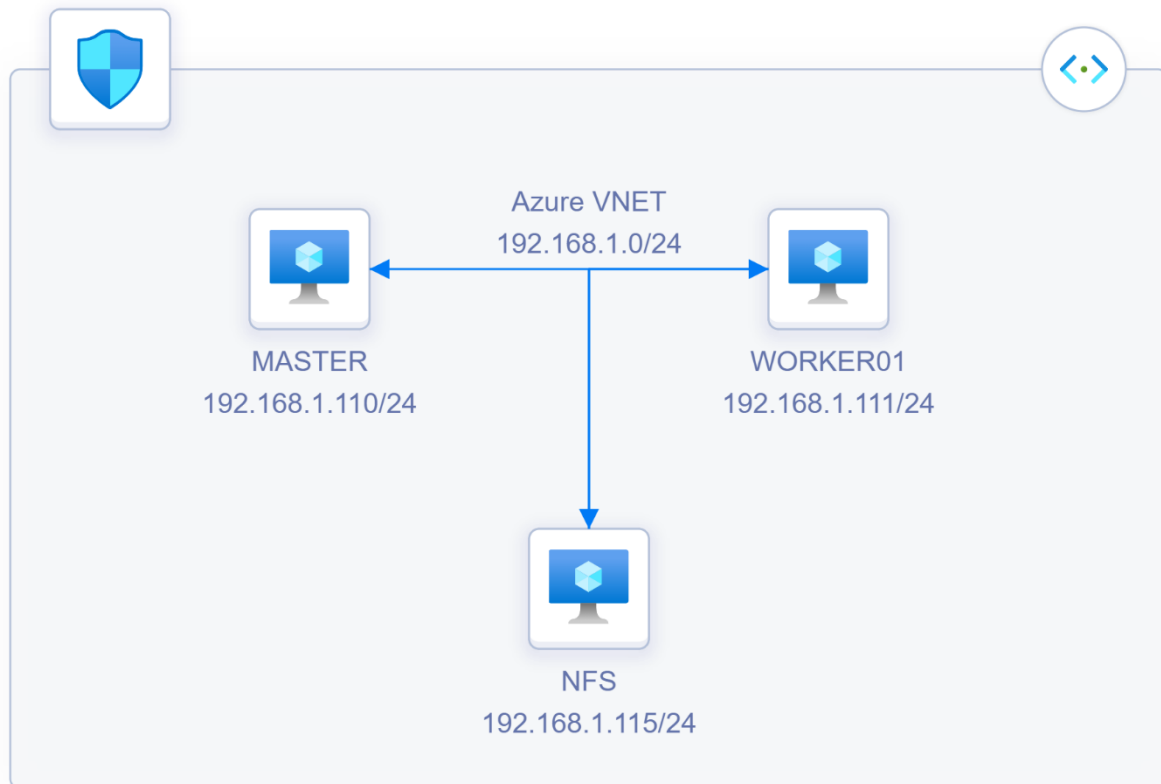
¿No encuentra la suscripción? [Mostrar más](#) ⓘ

i La suscripción no es válida para un aumento de cuota. Para solicitar un aumento de cuota, antes debe [actualizar a una suscripción de pago por uso](#) ⓘ. [Más información](#) ⓘ.

Siguiente

Puede verse en la anterior figura cómo dice que “La suscripción no es válida para un aumento de cuota”.

Por este problema o limitación se ha modificado el número de máquinas de la forma descrita anteriormente y que puede verse en el siguiente diagrama:



2. Procedimiento de despliegue

El repositorio donde está alojado todo el código se encuentra en GitHub y puede clonarse con la siguiente URL: <https://github.com/toninoes/devopsunirp2.git>

Una vez clonado presenta la siguiente estructura:

```
.
├── ansible
│   ├── 01-todos_hosts.yaml
│   ├── 02-servidor_nfs.yaml
│   ├── 03-master_workers.yaml
│   ├── 04-master.yaml
│   ├── 05-workers.yaml
│   ├── 06-guardar_token_master.yaml
│   ├── 07-unir_workers_cluster.yaml
│   ├── 08-despliegue_aplicacion.yaml
│   ├── comandos_linux_instalacion_k8b.txt
│   ├── deploy.sh
│   ├── group_vars
│   │   └── ips.yaml
│   ├── hosts
│   ├── roles
│   │   ├── all
│   │   │   ├── files
│   │   │   │   └── hosts
│   │   │   ├── tasks
│   │   │   │   └── main.yaml
│   │   ├── app
│   │   │   ├── files
│   │   │   │   └── jenkins.yaml
│   │   │   ├── tasks
│   │   │   │   └── main.yaml
│   │   ├── master
│   │   │   └── tasks
│   │   │       └── main.yaml
│   │   ├── master-workers
│   │   │   ├── files
│   │   │   │   ├── k8s.conf
│   │   │   │   └── kubernetes.repo
│   │   │   ├── tasks
│   │   │   │   └── main.yaml
│   │   ├── nfs-server
│   │   │   ├── files
│   │   │   │   └── exports
│   │   │   ├── tasks
│   │   │   │   └── main.yaml
│   │   └── workers
│   │       └── tasks
│   │           └── main.yaml
│   └── token_variable
├── LICENSE
├── README.md
├── terraform
│   ├── correccion-vars.tf
│   ├── main.tf
│   ├── network.tf
│   ├── no_deja_aumentar_cuota_Azure_Estudiantes.png
│   ├── security.tf
│   ├── vars.tf
│   └── vm.tf
└── 20 directories, 33 files
```

Es decir, nos encontramos dos directorios, uno con el plan de Terraform para desplegar la infraestructura de Azure y otro directorio para desplegar el clúster de Kubernetes con Ansible y la aplicación (Jenkins).

También nos encontramos con el correspondiente **README.md** donde hay una serie de indicaciones parecidas a las de este documento, aunque más resumidas y al mismo nivel nos encontraremos con la **licencia**, la cual se ha elegido para este propósito la **GNU General Public License v3.0**, la cual otorga los siguientes permisos:

- El material licenciado y sus derivados pueden utilizarse con fines comerciales.
- Puede modificarse.
- También puede ser distribuido.
- Esta licencia proporciona una concesión expresa de derechos de patente por parte de los contribuyentes.
- Se puede utilizar y modificar en privado.

Pero tiene las siguientes limitaciones:

- Esta licencia incluye una limitación de responsabilidad.
- Esta licencia establece explícitamente que NO proporciona ninguna garantía.

Y por tanto existen una serie de condiciones:

- Se debe incluir una copia de la licencia y el aviso de derechos de autor con el material con licencia.
- Los cambios realizados en el material con licencia deben documentarse.
- El código fuente debe estar disponible cuando se distribuya el material con licencia.
- Las modificaciones deben publicarse bajo la misma licencia al distribuir el material con licencia. En algunos casos, se puede utilizar una licencia similar o relacionada.

2.1. Despliegue de la infraestructura

Una vez visto en el apartado anterior la estructura de directorios del proyecto, para el despliegue de la infraestructura nos situaremos en el directorio **terraform** de dicho repositorio, donde se encuentra lo necesario para desplegar toda la infraestructura en Azure.

Pero antes debemos tener un par de claves en nuestro equipo, ya que se copiarán a las máquinas virtuales Azure en el despliegue y las utilizaremos también luego para el nodo master que hará de controller de ansible, para ello hacemos en nuestro equipo:

```
toni@tonipc:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/toni/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa
Your public key has been saved in id_rsa.pub
The key fingerprint is:
SHA256:d6ePc0yE/+ZhkgTgxPq345n4iEV5vmbUnCUft0YXPPUc tonipc
The key's randomart image is:
+---[RSA 4096]---+
|      o+..o o. |
|      oo.o o E|
|      ..o. .o..|
|      . o . .o .+o|
|      . o S B ++. o|
|      o  =.++  |
|      . +  += + |
|      . = . .o= +|
|      +.+  .o.o.|
+----[SHA256]-----+
```

También necesitaremos que se encuentre dentro de ese directorio el fichero de credenciales **credentials.tf** que tiene la siguiente estructura:

```
provider "azurerm" {
  features {}
  subscription_id = "<SUBSCRIPCION ID>"
  client_id       = "<APP_ID>"
  client_secret   = "<PASSWORD>"
  tenant_id       = "<TENANT>"
}
```

Estos datos se obtendrán al hacer **az login** con el cli de Azure.

También, como en esta práctica vamos a usar Centos8, deberás aceptar los términos de uso de dicha imagen en Azure con el siguiente comando:

```
[toni@tonipc: ~]# az vm image terms accept --urn cognosys:centos-8-stream-free:centos-8-stream-free:1.2019.0810
```

Todo esto lo puedes hacer directamente en la Cloud Shell de la consola de Azure, si no se quiere instalar el cliente en local.

Debes de disponer de la última versión de Terraform instalada, en la fecha de la escritura de este documento se hizo con la versión 1.0.1, aunque ya estaba disponible la 1.0.2 como puede verse:

```
toni@tonipc:~$ terraform --version
Terraform v1.0.1
on linux_amd64

Your version of Terraform is out of date! The latest version
is 1.0.2. You can update by downloading from https://www.terraform.io/downloads.html
```

...y finalmente ejecutar los siguientes comandos dentro del directorio **terraform**:

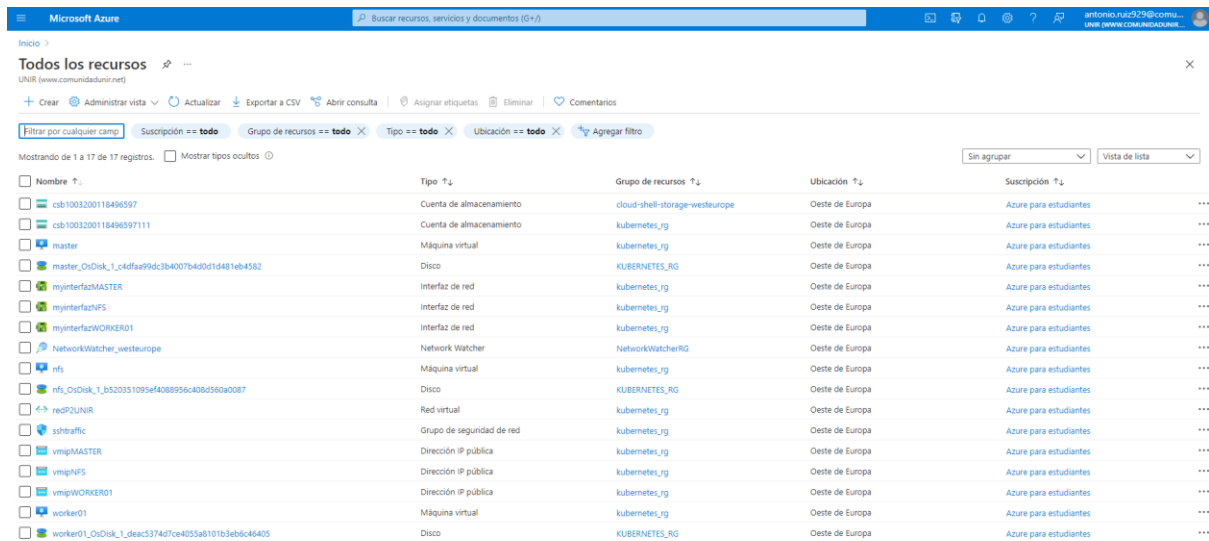
```
toni@tonipc:~/devopsunirp2/terraform$ terraform init
toni@tonipc:~/devopsunirp2/terraform$ terraform plan
toni@tonipc:~/devopsunirp2/terraform$ terraform apply
...
...
Plan: 17 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

azurerm_resource_group.rg: Creating...
azurerm_resource_group.rg: Creation complete after 1s [id=/subscriptions/6805d1d6-081c-49a6-ab0d-142248d48838/resourceGroups/kubernetes_rg]
azurerm_network_security_group.mySecGroup: Creating...
azurerm_public_ip.ipPublicaNFS: Creating...
azurerm_public_ip.ipPublicaWORKER01: Creating...
azurerm_virtual_network.miRed: Creating...
azurerm_public_ip.ipPublicaMASTER: Creating...
azurerm_storage_account.stAccount: Creating...
...
...
Apply complete! Resources: 17 added, 0 changed, 0 destroyed.
```

Al hacer **terraform apply**, se nos pedirá confirmación antes de desplegar, escribimos **yes** y tras unos pocos minutos podremos ver la infraestructura creada en nuestra cuenta de Azure:

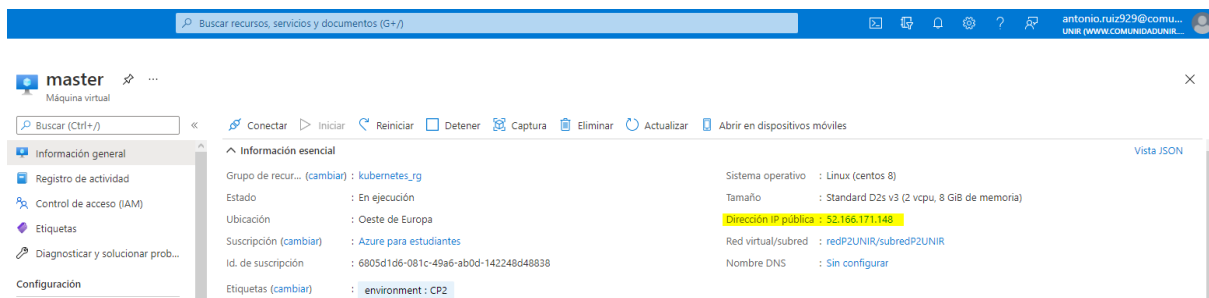


Nombre	Tipo	Grupo de recursos	Ubicación	Suscripción
csb1003200118496597	Cuenta de almacenamiento	cloud-shell-storage-westurope	Oeste de Europa	Azure para estudiantes
csb1003200118496597111	Cuenta de almacenamiento	kubernetes_rg	Oeste de Europa	Azure para estudiantes
master	Máquina virtual	kubernetes_rg	Oeste de Europa	Azure para estudiantes
master_OsDisk_1_c4dffa99dc3b4007b40d1d481eb4582	Disco	KUBERNETES_RG	Oeste de Europa	Azure para estudiantes
myinterfazMASTER	Interfaz de red	kubernetes_rg	Oeste de Europa	Azure para estudiantes
myinterfazNFS	Interfaz de red	kubernetes_rg	Oeste de Europa	Azure para estudiantes
myinterfazWORKER01	Interfaz de red	kubernetes_rg	Oeste de Europa	Azure para estudiantes
NetworkWatcher_westeurope	Network Watcher	NetworkWatcherRG	Oeste de Europa	Azure para estudiantes
nfs	Máquina virtual	kubernetes_rg	Oeste de Europa	Azure para estudiantes
nfs_OsDisk_1_b520351095ef408956c406d590a0087	Disco	KUBERNETES_RG	Oeste de Europa	Azure para estudiantes
redP2UNIR	Red virtual	kubernetes_rg	Oeste de Europa	Azure para estudiantes
sshtraffic	Grupo de seguridad de red	kubernetes_rg	Oeste de Europa	Azure para estudiantes
vmipMASTER	Dirección IP pública	kubernetes_rg	Oeste de Europa	Azure para estudiantes
vmipNFS	Dirección IP pública	kubernetes_rg	Oeste de Europa	Azure para estudiantes
vmipWORKER01	Dirección IP pública	kubernetes_rg	Oeste de Europa	Azure para estudiantes
worker01	Máquina virtual	kubernetes_rg	Oeste de Europa	Azure para estudiantes
worker01_OsDisk_1_deac5374d7ce4055a8101b3eb6c46405	Disco	KUBERNETES_RG	Oeste de Europa	Azure para estudiantes

2.2. Despliegue de kubernetes

Pasamos a indicar cómo desplegar kubernetes en la infraestructura desplegada anteriormente, esto se hará con Ansible (se utilizó la versión 2.9.6).

El directorio **ansible** contiene todos los ficheros necesarios para desplegar el clúster de Kubernetes, por tanto, una vez desplegada toda la infraestructura con Terraform, vamos a conectarnos al nodo master (por ejemplo, o aquel que elijamos como controller) por ssh el cual será desde donde lanzaremos los comandos de ansible, iremos primero a la consola de Azure y nos fijaremos en la IP pública de dicha máquina virtual:



Información esencial	
Grupo de recur...	kubernetes_rg
Estado	En ejecución
Ubicación	Oeste de Europa
Suscripción	Azure para estudiantes
Id. de suscripción	6805d1d6-081c-49a6-ab0d-142248d48838
Etiquetas	environment: CP2
Sistema operativo	Linux (centos 8)
Tamaño	Standard D2s v3 (2 vcpu, 8 GiB de memoria)
Dirección IP pública	52.166.171.148
Red virtual/subred	redP2UNIR/subredP2UNIR
Nombre DNS	Sin configurar

En este caso es la IP: 52.166.171.148, por lo tanto haremos desde nuestro equipo local:

```
toni@tonipc:~$ IP=52.166.171.148
toni@tonipc:~$ scp ~/.ssh/id_rsa ~/.ssh/id_rsa.pub adminUsername@$IP:~/.ssh
toni@tonipc:~$ ssh adminUsername@$IP
```

```

  ____
 |  _ \ | |  _ \ | |  _ \ | |  _ \
 | |_) | | | |_) | | | |_) | | | |_) |
 |  _ < | | |  _ < | | |  _ < | | |  _ <
 |_| \_> |_|_|_| \_> |_|_|_| \_> |_|_|_| \_>

```

```
Live Support @ https://www.secureanyccloud.com
AUTHORIZED USE ONLY.
Welcome! Support Email support@secureanyccloud.com
Last login: Mon Jul 12 16:31:12 2021 from 192.168.1.110
[adminUsername@master ~]$
```

Con esto ya habremos conectado con la máquina que hará de controller de Ansible y ya podremos continuar con los siguientes pasos, los cuales son:

- Instalar Ansible y Git
- Cambiar configuración de Ansible para que no compruebe la key de host al hacer SSH (para comodidad nuestra...
- Clonar el repositorio
- Situarnos en el directorio ansible
- Ejecutar el script deploy.sh, el cual contiene todos los playbook necesarios para el despliegue de kubernetes.

```
[adminUsername@master ~]$ sudo yum install epel-release -y
[adminUsername@master ~]$ sudo yum install ansible git -y
[adminUsername@master ~]$ sudo sed 's/^#//g' -i /etc/ansible/ansible.cfg
[adminUsername@master ~]$ git clone https://github.com/toninoes/devopsunirp2.git
[adminUsername@master ~]$ cd devopsunirp2/ansible/
[adminUsername@master ~]$ ./deploy.sh
PLAY [Hacerlos en todos los hosts] *****

TASK [Gathering Facts] *****
ok: [worker01]
ok: [nfs]
ok: [master]

TASK [all : Antes de actualizar todas las máquinas] *****
changed: [nfs]
changed: [worker01]
changed: [master]
...
...
PLAY [Hacerlo en master]
*****
*****

TASK [Gathering Facts]
*****
*****
ok: [master]

TASK [app : Copiar fichero de la aplicacion]
*****
*****
changed: [master]

TASK [Deploy application]
*****
*****
```

```

changed: [master]

PLAY RECAP
*****
*****
master      : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

```

...y tras un buen rato estará todo desplegado. Digo lo de rato porque en el update de los repositorios se invertirá alrededor de los 20 minutos, momento adecuado para prepararse un café...

Podemos ver cuando finalice cómo está el worker01 (sólo tenemos 1) en **Ready**:

```

[root@master ~]# kubectl get nodes
NAME        STATUS    ROLES                  AGE      VERSION
master      Ready     control-plane,master   3m23s    v1.21.2
worker01    Ready     <none>                 2m48s    v1.21.2

[root@master ~]#
[root@master ~]#
[root@master ~]# kubectl get pods -A -o wide
NAMESPACE   NAME                                     READY   STATUS    RESTARTS   AGE   IP
NODE        NOMINATED NODE   READINESS GATES
calico-system      calico-kube-controllers-7f58dbcbbd-7jmnh  1/1     Running   0          4m38s
192.169.219.65     master   <none>                 <none>
calico-system      calico-node-6lf5x                          1/1     Running   0          4m38s  192.168.1.110
master             <none>                                     <none>
calico-system      calico-node-btdzz                          1/1     Running   0          4m31s  192.168.1.111
worker01           <none>                                     <none>
calico-system      calico-typha-699db555f7-ngfdp             1/1     Running   0          4m39s  192.168.1.110
master             <none>                                     <none>
calico-system      calico-typha-699db555f7-tjdpt             1/1     Running   0          4m22s  192.168.1.111
worker01           <none>                                     <none>
default            jenkins-74c7d654c9-vz5kq                 1/1     Running   0          4m22s  192.169.5.1
worker01           <none>                                     <none>
haproxy-controller haproxy-ingress-65c5db48c8-k79xm          1/1     Running   0          4m48s  192.169.5.5
worker01           <none>                                     <none>
haproxy-controller ingress-default-backend-78f5cc7d4c-qrd2g  1/1     Running   0          4m49s  192.169.5.3
worker01           <none>                                     <none>
kube-system        coredns-558bd4d5db-2lgn                   1/1     Running   0          4m50s  192.169.5.4
worker01           <none>                                     <none>
kube-system        coredns-558bd4d5db-hcjss                  1/1     Running   0          4m50s  192.169.5.2
worker01           <none>                                     <none>
kube-system        etcd-master                              1/1     Running   0          5m3s   192.168.1.110
master             <none>                                     <none>
kube-system        kube-apiserver-master                     1/1     Running   0          5m3s   192.168.1.110
master             <none>                                     <none>
kube-system        kube-controller-manager-master            1/1     Running   0          5m3s   192.168.1.110
master             <none>                                     <none>
kube-system        kube-proxy-dzvt9                          1/1     Running   0          4m50s  192.168.1.110
master             <none>                                     <none>
kube-system        kube-proxy-vtf8t                          1/1     Running   0          4m31s  192.168.1.111
worker01           <none>                                     <none>
kube-system        kube-scheduler-master                     1/1     Running   0          5m3s   192.168.1.110
master             <none>                                     <none>
tigera-operator    tigera-operator-86c4fc874f-kq2rw          1/1     Running   0          4m50s  192.168.1.110
master             <none>                                     <none>

```

Finalmente vemos también los servicios:

```
[root@master ~]#  
[root@master ~]# kubectl get svc -A  
NAMESPACE          NAME                                     TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)  
AGE  
calico-system       calico-kube-controllers-metrics        ClusterIP  10.105.104.93    <none>           9094/TCP  
5m32s  
calico-system       calico-typha                           ClusterIP  10.101.101.111   <none>           5473/TCP  
6m30s  
default             kubernetes                             ClusterIP  10.96.0.1        <none>           443/TCP  
6m57s  
haproxy-controller  haproxy-ingress                        NodePort   10.110.165.58    <none>           80:32645/TCP,443:32153/TCP,1024:32132/TCP 6m39s  
haproxy-controller  ingress-default-backend                ClusterIP  10.102.49.197    <none>           8080/TCP  
6m40s  
kube-system         kube-dns                               ClusterIP  10.96.0.10       <none>           53/UDP,53/TCP,9153/TCP 6m55s  
[root@master ~]#
```

Según lo anterior tenemos:

- El puerto del host 32645 se encuentra mapeado al 80 de los contenedores.
- El puerto del host 32153 se encuentra mapeado al 443 de los contenedores.
- El puerto del host 32132 se encuentra mapeado al 1024 de los contenedores. Que tal y como se comentó en los laboratorios de Kubernetes del curso, este puerto se utiliza para estadísticas de haproxy.

3. Despliegue de la aplicación

La aplicación elegida para desplegar es Jenkins, la cual puede verse en DockerHub con imagen oficial <https://hub.docker.com/r/jenkins/jenkins>

Como ya se ha visto en el presente curso, Jenkins ayuda en la automatización de parte del proceso de desarrollo de software mediante integración continua y facilita también la entrega continua, admitiendo herramientas de control de versiones como Subversion, Git, Mercurial, etc...

3.1. Despliegue de la aplicación

Muestro a continuación el fichero que se utilizará para el despliegue y dónde se encuentra dentro del repositorio:

Fichero: devopsunirp2/ansible/roles/app/files/jenkins.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jenkins
  labels:
    app: jenkins
spec:
  replicas: 1
  selector:
    matchLabels:
      app: jenkins
  template:
    metadata:
      labels:
        app: jenkins
    spec:
      containers:
        - name: jenkins
          image: jenkins/jenkins:lts
          ports:
            - containerPort: 80
```

Este fichero se utiliza en la última etapa del **deploy.sh**, es decir, el que se marca en negrita al final:

Fichero: devopsunirp2/ansible/deploy.sh

```
#!/bin/bash

# añadir tantas líneas como sean necesarias para el correcto despliegue
ansible-playbook -i hosts 01-todos_hosts.yaml
ansible-playbook -i hosts 02-servidor_nfs.yaml
ansible-playbook -i hosts 03-master_workers.yaml
ansible-playbook -i hosts 04-master.yaml
ansible-playbook -i hosts 05-workers.yaml
ansible-playbook -i hosts 06-guardar_token_master.yaml
ansible-playbook -i hosts 07-unir_workers_cluster.yaml
ansible-playbook -i hosts 08-despliegue_aplicacion.yaml
```

Después de crear el deployment podemos ver el estado del POD:

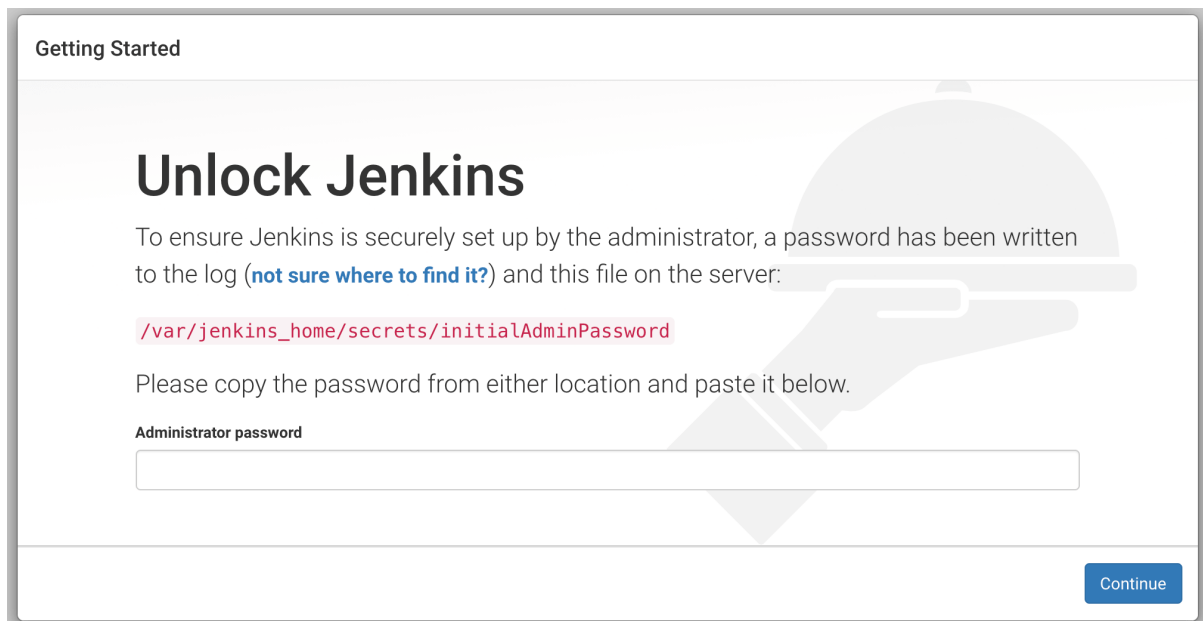
```
[root@master ~]# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
jenkins-74c7d654c9-vz5kq          1/1     Running   0           48m
[root@master ~]#
[root@master ~]#
[root@master ~]#
[root@master ~]#
[root@master ~]# kubectl describe pod jenkins-74c7d654c9-vz5kq
Name:                               jenkins-74c7d654c9-vz5kq
Namespace:                          default
Priority:                             0
Node:                               worker01/192.168.1.111
Start Time:                         Mon, 12 Jul 2021 18:57:43 +0200
Labels:                             app=jenkins
Annotations:                         cni.projectcalico.org/podIP: 192.169.5.1/32
                                    cni.projectcalico.org/podIPs: 192.169.5.1/32
Status:                             Running
IP:                                 192.169.5.1
IPs:
  IP:                                192.169.5.1
Controlled By:                      ReplicaSet/jenkins-74c7d654c9
Containers:
  jenkins:
    Container ID:                    docker://4ff969cd8afed4b4f054f9625bcfeaaee3de8806f4915a49668529ed0d3c4f1b3
    Image:                           jenkins/jenkins:lts
    Image ID:                         docker-
    pullable://jenkins/jenkins@sha256:f430bcb70ab64af5e96a13427ee2e3c8ef1c1cba3688c6341ee2486a91deba2a
    Port:                             80/TCP
    Host Port:                        0/TCP
    State:                           Running
      Started:                       Mon, 12 Jul 2021 18:58:26 +0200
    Ready:                           True
    Restart Count:                    0
    Environment:                     <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-tm6n2 (ro)
Conditions:
  Type             Status
  Initialized      True
  Ready            True
  ContainersReady  True
  PodScheduled     True
Volumes:
  kube-api-access-tm6n2:
    Type:              Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName:       kube-root-ca.crt
    ConfigMapOptional:    <nil>
    DownwardAPI:         true
QoS Class:           BestEffort
Node-Selectors:      <none>
Tolerations:         node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                     node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason             Age          From          Message
  ----     -
  Warning   FailedScheduling   48m (x4 over 49m)  default-scheduler  0/2 nodes are available: 1 node(s) had taint {node-role.kubernetes.io/master: }, that the pod didn't tolerate, 1 node(s) had taint {node.kubernetes.io/not-ready: }, that the pod didn't tolerate.
  Normal    Scheduled          48m          default-scheduler  Successfully assigned default/jenkins-74c7d654c9-vz5kq to worker01
  Normal    Pulling            48m          kubelet         Pulling image "jenkins/jenkins:lts"
  Normal    Pulled             48m          kubelet         Successfully pulled image "jenkins/jenkins:lts"
in 28.767090808s
  Normal    Created            47m          kubelet         Created container jenkins
  Normal    Started            47m          kubelet         Started container jenkins
```


También podemos ejecutar una shell en el contenedor:

```
[root@master ~]# kubectl exec -ti jenkins-74c7d654c9-vz5kq -- /bin/bash
jenkins@jenkins-74c7d654c9-vz5kq:/$
jenkins@jenkins-74c7d654c9-vz5kq:/$ cat /var/jenkins_home/secrets/initialAdminPassword
8055306f24964dedb3783221495cf4e1
jenkins@jenkins-74c7d654c9-vz5kq:/$
```

3.2. Acceso a la aplicación

Podemos ver en el anterior apartado el password para proceder a la instalación, ubicado dentro del contenedor en `(/var/jenkins_home/secrets/initialAdminPassword)`, la cual es **8055306f24964dedb3783221495cf4e1** y que habría que introducir en una pantalla como esta:



Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/jenkins_home/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

Continue

...pero que no me ha sido posible llegar a ella por ninguna de las maneras por más que lo he intentado. Lo detallo en el capítulo siguiente.

4. Problemas encontrados

- No he sido capaz de acceder a la aplicación desplegada (Jenkins) desde fuera del clúster poniendo la IP pública de master y el puerto al que está mapeado el puerto 80 en los contenedores, el 32645, es decir no me ha funcionado <http://52.166.171.148:32645/jenkins> ni <http://52.166.171.148:32645> no he sido capaz.

He abierto el puerto de entrada a master 32645 y tampoco:

myinterfazMASTER

Configuración de IP: 

configuracionMASTER (Principal) 

 **Interfaz de red: myinterfazMASTER** [Reglas de seguridad vigentes](#) [Solucionar problemas de conexión de VM](#) [Topología](#)

Red virtual/subred: redP2UNIR/subredIP2UNIR **IP pública de NIC: 52.166.171.148** IP privada de NIC: 192.168.1.110 Redes aceleradas: **Deshabilitado**

Reglas de puerto de entrada [Reglas de puerto de salida](#) [Grupos de seguridad de aplicación](#) [Equilibrio de carga](#)

 Grupo de seguridad de red: sshtraffic (se conectó a la interfaz de red: myinterfazMASTER)
Impactos 0 subredes, 3 interfaces de red

[Agregar regla de puerto de entrada](#)

Prioridad	Nombre	Puerto	Protocolo	Origen	Destino	Acción	
1001	 SSH	22	TCP	Cualquiera	Cualquiera	 Permitir	...
1011	 jenkins	32645	Cualquiera	Cualquiera	Cualquiera	 Permitir	...
65000	AllowVnetInBound	Cualquiera	Cualquiera	VirtualNetwork	VirtualNetwork	 Permitir	...
65001	AllowAzureLoadBalancerInBound	Cualquiera	Cualquiera	AzureLoadBalancer	Cualquiera	 Permitir	...
65500	DenyAllInBound	Cualquiera	Cualquiera	Cualquiera	Cualquiera	 Denegar	...

5. Referencias

[1] **Documentación de Azure:** <https://docs.microsoft.com/es-es/azure/?product=featured>

[2] **Ansible Documentation:** <https://docs.ansible.com/ansible/latest/index.html>

[3] **Documentación de Kubernetes:** <https://kubernetes.io/es/docs/home/>

[4] **Documentación de Terraform:** <https://www.terraform.io/docs/index.html>

[5] **Repositorio GitHub curso Jadebustos:** <https://github.com/jadebustos/devopslabs.git>