Fuente: Bing Chat AI

Bloque 6. Terraform

MÁSTER EN INGENIERÍA WEB 2023 – 2024

Administración de Sistemas Operativos v1.2 *("L-62 Princesa de Asturias")*

© José Manuel Redondo López. Universidad de Oviedo

Contenido

Nivel 1 (Cabo): Instalación y uso básico

¿Qué necesito para hacer este bloque de actividades?

Para realizar este bloque de actividades necesitas una máquina física o virtual *Linux*, **como por ejemplo las que se puedan haber creado con el módulo de *Vagrant***. Se ha probado sobre *Ubuntu 18.04*). Necesitas:

- **Un software de virtualización instalado**: Se recomienda *VirtualBox*, como en el caso de *Vagrant*.
- **Núcleos de CPU**: Con 2 se puede trabajar perfectamente, un extra siempre es bienvenido
- **RAM**: Mínimo 2, mejor 4 o más
- **SO**: Un *Linux*; se ha probado sobre *Ubuntu 18.04* con éxito.

Debido a la naturaleza de *Terraform*, **este módulo además tiene un alto grado de interacción con el resto de la asignatura** (especialmente con el bloque de contenedores de aplicaciones), por lo que se recomienda ver el principio de cada ejercicio para **saber si es necesario haber hecho un nivel mínimo en otros módulos para realizarlo correctamente**.

N1TERRA_INSTALAR: Instalación de *Terraform* *(0,1 puntos)*

Descripción de la actividad

Consiste en **tener una instalación operativa de *Terraform*** que funcione en local **con un proveedor *Docker***.

(NOTA: Este ejercicio requiere haber hecho al menos la instalación de Docker y Docker Compose del Nivel 1 del boletín de ejercicios de contenedores de aplicaciones)

Resultados esperados

Eres capaz de instalar correctamente *Terraform* para que funcione integrado con *Docker*

Información necesaria para su realización

Esta actividad consiste en la puesta en marcha de una instalación de *Terraform* sobre un sistema operativo que lo soporte, como *Ubuntu,* **usando *Docker como proveedor***. Es necesaria por tanto la instalación en la misma máquina de *Docker*, tal y como se indica en el boletín de ejercicios correspondiente.

NOTA: Para otros ejercicios se necesitará además *LXD*, *Vagrant* o *Kubernetes*, pero eso se indicará en los ejercicios correspondientes y no es necesario instalar estos tres últimos ahora, puesto que dependerá de si los haces o no.

N1TERRA_DESPDOCKERHUB: Desplegar una imagen del *Docker Hub* mediante *Terraform* *(0,2 puntos)*

Descripción de la actividad

Consiste en aprender a **desplegar imágenes *Docker* del *Docker Hub* usando *Terraform*** y su forma de trabajo habitual.

(NOTA: Este ejercicio requiere haber hecho al menos el Nivel 1 del bloque de contenedores de aplicaciones)

Resultados esperados

Eres capaz de desplegar, **validando la configuración previamente**, una imagen del *Docker Hub* con tu instalación local de *Terraform* . Además, entiendes el propósito del **plan de ejecución**, modificando un despliegue inicial y viendo qué ocurre con el plan tras dicha modificación. Puedes contestar a estas preguntas:

- *¿Cuál es el procedimiento estándar para lidiar con un despliegue de Terrafom, de forma que podamos estar seguros de lo que va a pasar antes de que pase?*
- *A tenor de lo que has visto, ¿Crees que Terraform facilita la modificación dinámica de las infraestructuras que controla?*

Información necesaria para su realización

Esta actividad consiste en desplegar una imagen *Docker* del *Docker Hub* de tu elección con *Terraform* y comprobar que funciona, ejecutando los procedimientos de validación explicados en teoría para desplegar (`init` - `plan` - `apply`) antes de hacerlo, para que así puedas visualizar el plan de despliegue antes de proceder a aplicarlo.

Para hacerlo, desplegar primero la imagen `ubuntu:latest` y comprobar qué ocurre. Cambiar luego el mismo despliegue para que se haga con la imagen `ubuntu/apache2:latest` y ver el contenido del nuevo plan de ejecución para ver qué indica ahora y las conclusiones que puedes sacar de lo que ves.

N1TERRA_ DESPDOCKERLOCAL: Desplegar un contenedor de una imagen local mediante *Terraform* *(0,1 puntos)*

Descripción de la actividad

Consiste en aprender a **desplegar imágenes *Docker* que tengas tú creadas localmente usando *Terraform*** y su forma de trabajo habitual.

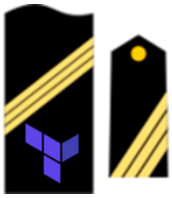
(NOTA: Este ejercicio requiere haber hecho al menos el Nivel 2 del bloque de contenedores de aplicaciones)

Resultados esperados

Eres capaz de desplegar, validando la configuración previamente, una imagen de *Docker* creada por ti con tu instalación local de *Terraform*

Información necesaria para su realización

Esta actividad consiste en **desplegar una imagen *Docker* que hayas hecho tú mismo** (por ejemplo en alguna actividad del módulo de *Docker*) con *Terraform* y comprobar que funciona, ejecutando los procedimientos de validación de teoría antes de hacerlo. También **debes visualizar el plan de despliegue** antes de proceder a realizarlo.



Nivel 2 (Sargento): Usos frecuentes de Terraform

N2TERRA_EXPSERVICIO: Exponer un servicio ofrecido por un contenedor mediante *Terraform* *(0,1 puntos)*

Descripción de la actividad

Consiste en **aprender a exponer un servicio** que se haya desplegado en un contenedor usando *Terraform*.

(NOTA: Este ejercicio requiere haber hecho al menos el Nivel 1 del bloque de contenedores de aplicaciones)

Resultados esperados

Eres capaz de ofertar cualquier servicio de un contenedor desplegado con *Terraform* a clientes externos

Información necesaria para su realización

En esta actividad debes exponer un servicio ofertado por un contenedor que hayas desplegado previamente con *Terraform* (por ejemplo de una actividad anterior), para que sea accesible desde fuera del *host*.

N2TERRA_GRAFOR: Visualizar un grafo de recursos *(0,1 puntos)*

Descripción de la actividad

Consiste en aprender a **generar y ver el grafo de recursos** de un despliegue de *Terraform*.

(NOTA: Este ejercicio requiere haber hecho al menos el Nivel 1 del bloque de contenedores de aplicaciones)

Resultados esperados

Eres capaz de volcar la planificación de un despliegue de *Terraform* en un fichero que puedas incluir en un informe de operaciones, por ejemplo. Puedes contestar a estas preguntas:

- *¿En qué situaciones crees que es más útil el grafo de recursos?*
- *¿Cómo crees que te puede ayudar en instalaciones que tengan muchos recursos en nube, incluso entre varios proveedores?*

Información necesaria para su realización

Esta actividad requiere **desplegar el grafo de recursos** de un despliegue de *Terraform* (sirve cualquiera de las actividades anteriores) y usar los contenidos de esta web: https://www.terraform.io/cli/commands/graph para poder **visualizarlo de forma gráfica** y con colores. Para usar el comando `dot` necesitas instalar el paquete `graphviz`.

N2TERRA_DESPLXD: Desplegar una imagen *LXD* mediante *Terraform* *(0,2 puntos)*

Descripción de la actividad

Consiste en **aprender a desplegar un contenedor LXD** mediante *Terraform*.

(NOTA: Este ejercicio requiere haber hecho al menos el Nivel 1 del bloque de contenedores de aplicaciones y del bloque de contenedores de SO)

Resultados esperados

Eres capaz de desplegar un contenedor *LXD* en tu instalación local del mismo gracias a *Terraform*. Puedes contestar a esta pregunta: *¿Cambia la forma de trabajar habitualmente con Terraform y sus despliegues por el hecho de que lo que se está desplegando es un contenedor LXD?*

Información necesaria para su realización

Si tenemos una instalación operativa de los contenedores de SO *LXD*, como la que vimos en el boletín de ejercicios correspondiente, **podemos usar *Terraform* para desplegar contenedores sobre ella** gracias al proveedor `terraform-lxd/lxd` (https://registry.terraform.io/providers/terraform-lxd/lxd/1.7.0/docs). Este proveedor **conecta con el *daemon LXD*** a través del *socket Unix* local que se crea cuando se instala. Usa la librería para clientes de LXD, que mira la configuración del *daemon* en `~/.config/lxc/` para autenticarse con él automáticamente sin que tengamos que hacer nada más.

Si en lugar de nuestra instalación local quisiéramos usar una instalación de LXD remota, el proveedor se comunicaría con ella vía HTTPS. No obstante, necesitaríamos una configuración extra: https://registry.terraform.io/providers/terraform-lxd/lxd/1.7.0/docs. No obstante, **en esta actividad solo se usará una instalación local**.

Como todo proveedor *Terraform*, éste **añade *resource types* y *data sources*** que modelan los conceptos que hemos visto en el bloque de contenedores de SO. Para hacer una prueba básica de la interacción *Terraform - LXD* se recomienda aplicar este fichero de despliegue con el procedimiento habitual, donde se puede ver cómo *Terraform* ahora entiende el tipo de recurso `lxd_container` y sus propiedades. **Por favor, no olvides hacer** `terraform init` para descargar e inicializar el proveedor *LXD* la primera vez que hagas un despliegue con él.

En este ejemplo que puedes usar como punto de partida se está usando una imagen de *Ubuntu 20.04LTS* para el despliegue, pero puedes usar cualquiera de las vistas en el bloque de ejercicios de contenedores de SO, con la misma nomenclatura.

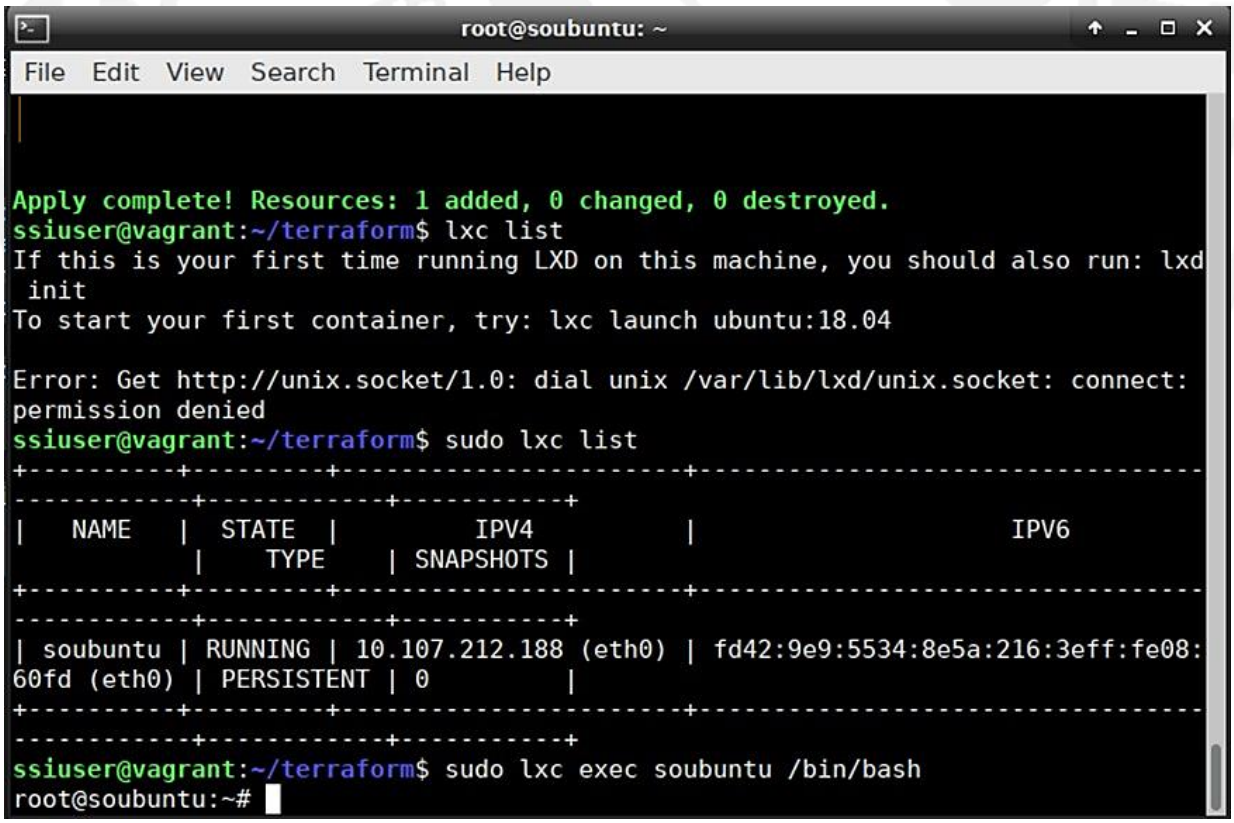
```
terraform {
  required_providers {
    lxd = {
      source = "terraform-lxd/lxd"
      version = "1.5.0"
    }
  }
}

resource "lxd_container" "soubuntu" {
  name      = "soubuntu"
  image     = "ubuntu:20.04"
  ephemeral = false

  config = {
    "boot.autostart" = true
  }

  limits = {
    cpu = 1
  }
}
```

Si todo es correcto, al final deberías ver este contenedor ejecutándose en LXD (o el nombre que hayas usado tú para tu despliegue).



Nivel 3 (Teniente de navío): Manejo de volúmenes y redes en contenedores con Terraform

N3TERRA_PRIVNETDOCKER: Crear una red privada entre contenedores desplegados en *Terraform* *(0,2 puntos)*

Descripción de la actividad

Consiste en aprender a **crear una red privada entre contenedores *Docker*** que se hayan desplegado con *Terraform*

(NOTA: Este ejercicio requiere haber hecho al menos el Nivel 1 del bloque de contenedores de aplicaciones)

Resultados esperados

Eres capaz de desplegar con *Terraform* una infraestructura de contenedores que puedan comunicarse entre sí.

Información necesaria para su realización

En esta actividad se trata de **desplegar dos o más contenedores *Docker*** de vuestra elección con *Terraform* siguiendo las explicaciones dadas en la teoría, de manera que **entre ambos exista una red local** que permita la comunicación entre ellos, probando además que dicha comunicación funciona y desplegando antes un grafo que muestre los recursos que van a crearse.

N3TERRA_VOLLXD: Usar *Terraform* para añadir un volumen a un contenedor *LXD* *(0,3 puntos)*

Descripción de la actividad

Consiste en **aprender a añadir un volumen a un contenedor LXD** gracias a *Terraform*.

(NOTA: Este ejercicio requiere haber hecho al menos el Nivel 1 del bloque de contenedores de aplicaciones y el Nivel 3 del bloque de contenedores de SO)

Resultados esperados

Eres capaz de conectar un nuevo volumen a un contenedor *LXD* desplegado con *Terraform* previamente, hacer un redespliegue y comprobar que el volumen está accesible. Puedes contestar a esta pregunta:

¿Entiendes cómo se pueden ir actualizando las definiciones de los contenedores con nuevos elementos gracias a las funcionalidades de Terraform?

Información necesaria para su realización

Esta actividad **es un complemento a la del nivel anterior que desplegaba una imagen LXD con *Terraform***. Se trata de **actualizar** ese despliegue **para añadirle un volumen** con los *resource types* que define el proveedor para ello que se describirán a continuación:

- `lxd_storage_pool`: Permite definir una nueva *storage pool* para guardar información en nuestros contenedores. La más fácil de crear es la **basada en un directorio**, que se guardará en la ruta por defecto para *storage pools* de LXD con un fichero de definición como este. Se podría crear uno ZFS como se vio en el módulo de LXD, pero no queremos hacer este ejercicio más complejo.

```
resource "lxd_storage_pool" "custompool" {
  name = "custompool"
  driver = "dir"
  config = {
    source = "/var/lib/lxd/storage-pools/custompool"
  }
}
```

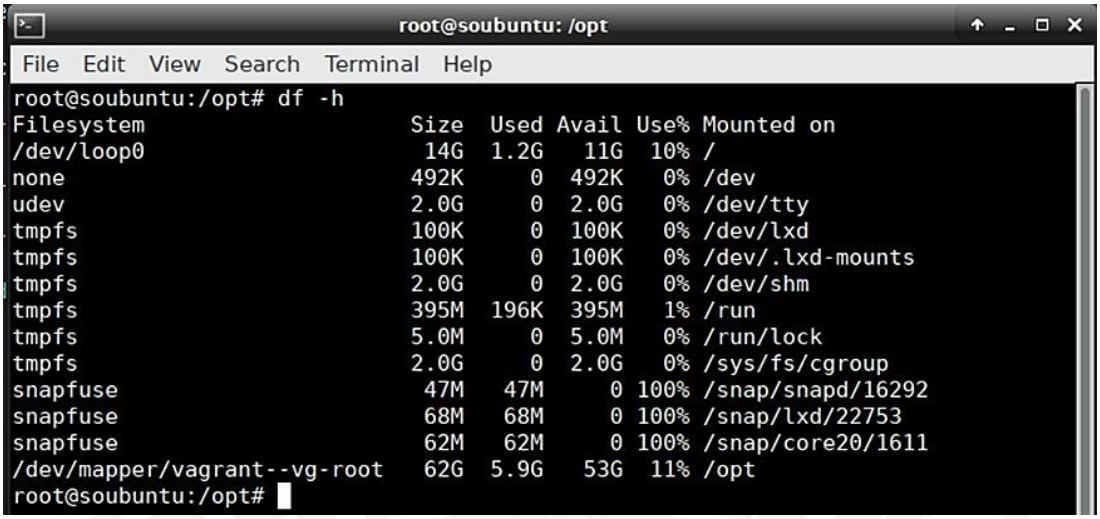
- `lxd_volume`: Crea la definición de un volumen usable por contenedores, conectándolo con la *storage pool* que guardará su información. Fíjate como se está usando el nombre que le hemos puesto a la *pool* anterior para hacer esa conexión.

```
resource "lxd_volume" "volumenprueba" {
  name = "myvolume"
  pool = "${lxd_storage_pool.custompool.name}"
}
```

Ahora **amplía el recurso** `lxd_container` anterior para añadirle un nuevo dispositivo (`device`) **dentro de su definición** que conecte el volumen a una ruta del disco del contenedor, usando este código como guía, donde se usa el nombre del volumen y de la *pool* anterior para hacer la conexión con el contenedor:

```
  device {
    name = "volumenprueba"
    type = "disk"
    properties = {
      path = "/opt/"
      source = "${lxd_volume.volumenprueba.name}"
      pool = "${lxd_storage_pool.custompool.name}"
    }
  }
```

Vuelve a desplegar la infraestructura en *Terraform* atendiendo especialmente al plan propuesto de cambios, donde se ve lo que se ve a añadir a la existente. Entra dentro del contenedor desplegado y comprueba con `df -h` que hay un volumen conectado a la ruta especificada (`/opt` en el ejemplo, como se ve en la imagen).



N3TERRA_PROFLXD: Usar *Terraform* para añadir una tarjeta de red a un contenedor *LXD* usando varios perfiles *(0,4 puntos)*

Descripción de la actividad

Consiste en **aprender a manejar los perfiles LXD mediante *Terraform***, añadiendo una tarjeta de red a un contenedor.

(NOTA: Este ejercicio requiere haber hecho al menos el Nivel 1 del bloque de contenedores de aplicaciones y el Nivel 4 del bloque de contenedores de SO)

Resultados esperados

Eres capaz de añadir una tarjeta de red operativa a un contenedor *LXD* creando un perfil que la defina y añadiendo dicho perfil al contenedor.

Información necesaria para su realización

Como se vio en el bloque de *LXD*, es posible **modificar la configuración de sus contenedores** mediante el uso de **perfiles**. Existe además un perfil por defecto, `default`, que le da a cada contenedor acceso a almacenamiento y una interfaz de red `eth0` que le permite conectarse a Internet.

En esta actividad se va a usar otro *resource type* del proveedor *LXD* descrito en el ejercicio anterior para añadirle al despliegue que usamos en dicho ejercicio **un nuevo perfil** que defina una nueva tarjeta de red tipo `macvlan`, que, como vimos, representa a una red privada. Esta tarjeta se llamará `eth1` y **en la máquina virtual de pruebas debe existir una tarjeta `eth1` también** que haga de "padre" de la misma.

Para hacer todo esto es necesario **añadir un nuevo tipo de recurso al fichero anterior**, llamado `lxd_network`, como este. Puedes ver más ejemplos de cómo añadir redes en esta enlace: https://registry.terraform.io/providers/terraform-lxd/lxd/latest/docs/resources/network

```
resource "lxd_network" "internal" {
  name = "internal"
```

```
  config = {
    "ipv4.address" = "192.168.200.1/24"
 #   "ipv4.nat"     = "true"
 #   "ipv6.address" = "fd42:474b:622d:259d::1/64"
 #   "ipv6.nat"     = "true"
  }
}
```

Hemos dejado comentadas algunas opciones de este recurso porque en el escenario planteado no van a usarse, pero sí tienen aplicación en caso de crear tarjetas de red `bridge`. Estos son `ipv4.nat`, que nos permite usar la tarjeta red padre para salir a redes externas, y su equivalente en IPv6, además de cómo asignar una dirección de esta clase a un recurso de red si lo necesitamos.

Una vez definida la red, podemos **crear un nuevo perfil** que especifique cómo la van a usar los contenedores a los que se aplique dicho perfil, **usando para ello un nuevo *resource type*** llamado `lxd_profile` que nos da el proveedor LXD para *Terraform*. Este trozo de código muestra cómo crearlo:

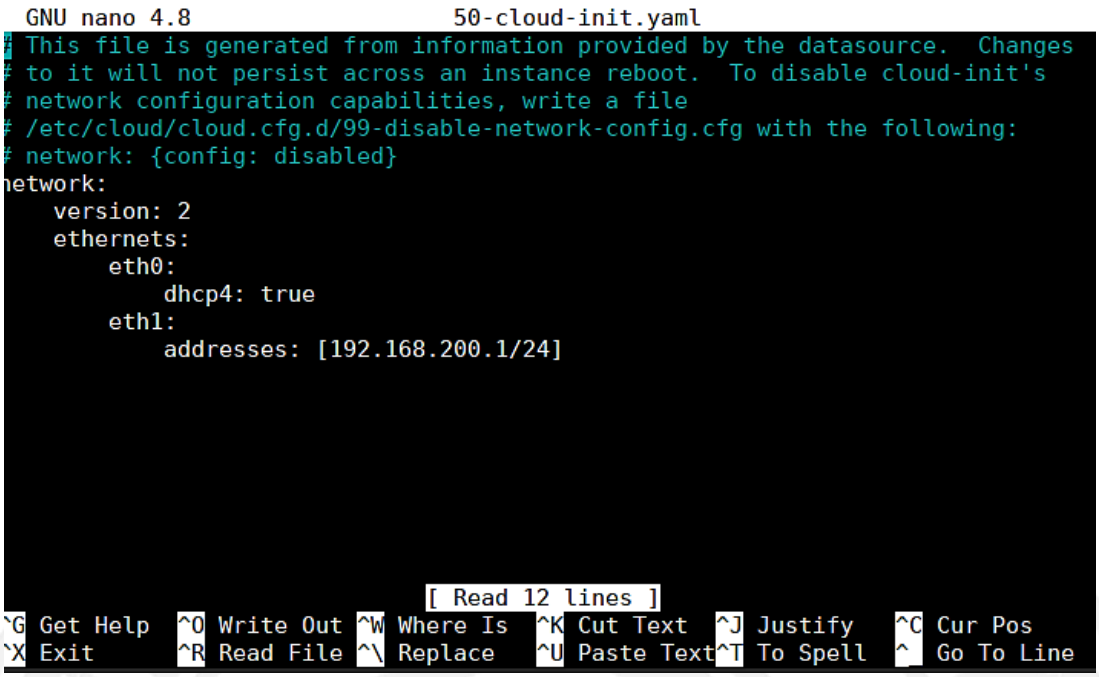
```
resource "lxd_profile" "profileinternallan1" {
  name = "profileinternallan1"
  device {
    name = "eth1"
    type = "nic"

    properties = {
      nictype = "macvlan"
      parent  = "${lxd_network.internal.name}"
    }
  }
}
```

Por último, solo nos falta decirle al recurso `lxd_container` anterior que ahora va a usar dos perfiles: el que ya usaba (el `default` que se aplica por defecto), y el nuevo que acabamos de crear. Para ello, basta con añadir esta línea dentro de la definición del `resource`.

```
  profiles = ["default", "${lxd_profile.profileinternallan1.name}"]
```

Si ahora redesplegamos como de costumbre y entramos dentro del contenedor, es posible que veamos que la tarjeta de red existe (`ip a`), **pero no tiene IP asignada**. Esto es por culpa de que no existe un servicio DHCP que se la asigne y que no se asigna por algún motivo la IP especificada en la definición del fichero *Terraform*. Por tanto, si te ocurre esto, debes hacerlo de manera manual como se hace en un *Linux* típico, modificando el fichero `/etc/netplan/50-cloud-init.yaml` del contenedor para que quede así:



Tras hacer un `sudo netplan apply` ahora ya deberíamos ver que `eth1` tiene una IP asignada dentro del contenedor, terminando esta actividad.

(***NOTA***: *Si hubiéramos tenido un cliente DHCP usable por la red de la tarjeta que acabamos de crear, se podría haber obtenido una IP automáticamente añadiendo un* ***provisioner*** *como éste al* `resource` `lxd_container` *del fichero. Esto* ***NO*** *es necesario hacerlo en esta actividad, porque no tenemos servidor DHCP que la tarjeta pueda usar, y sólo está aquí para que veáis como se haría el provisionamiento de una IP a un contenedor LXD con Terraform*)

```
  provisioner "local-exec" {
    command = "lxc exec local:${self.name} dhclient eth1"
  }
```

Nivel 4 (Capitán de navío): Despliegue de endpoints más complejos con Terraform

N4TERRA_VAGRANT: Despliegue de una máquina virtual *Vagrant* mediante *Terraform* *(0,3 puntos)*

Descripción de la actividad

Consiste en **aprender a desplegar máquinas virtuales *Vagrant*** usando *Terraform*.

(NOTA: Este ejercicio requiere haber hecho al menos el Nivel 1 del bloque de contenedores de aplicaciones y el Nivel 1 del bloque de Vagrant)

Resultados esperados

Eres capaz de desplegar con *Terraform* una máquina virtual definida en una *Vagrantfile* sobre una instalación de *Vagrant* operativa.

Información necesaria para su realización

Con *Terraform* también **podemos desplegar máquinas virtuales *Vagrant*** en una instalación de *Vagrant* operativa como la del bloque de ejercicios correspondiente de este curso. En este caso tener la infraestructura para hacerla funcionar conlleva ciertos problemas, puesto que en el caso de que se instale *Terraform* en una máquina virtual (caso típico) **la máquina virtual debe tener soporte para virtualización anidada** para que *Vagrant* pueda lanzar máquinas usando un proveedor típico, como *VirtualBox*.

Otra opción es hacer una instalación de *Terraform* para *Windows*, donde es más fácil instalar un *Vagrant* operativo por ser el *host*. En caso de no disponer de ninguna de estas dos opciones, podremos definir ficheros con despliegues y hacer `init` y `plan` con *Terraform*, pero la creación de la máquina fallará al hacer `apply`. **No obstante, para esta actividad sirve cualquiera de estas opciones, incluso si no se puede hacer `apply`**.

El despliegue de *Terraform* sobre *Vagrant* es posible gracias al *provider* `bmatcuk/vagrant` (https://registry.terraform.io/providers/bmatcuk/vagrant/latest/docs). De nuevo se definen los *resource types* y *data sources* que modelan todos los elementos de *Vagrant* vistos en el bloque correspondiente. Este *provider* no obstante funciona de una manera algo diferente al resto, porque **necesita que le demos la ruta de la `Vagrantfile` que define la máquina** (o máquinas) que vamos a desplegar. Por tanto, podemos desplegar con *Terraform* cualquier máquina *Vagrant* que hayamos definido en el bloque de ejercicios correspondiente. Para ello podemos usar un fichero como este:

```
terraform {
  required_providers {
```

```
    vagrant = {
      source  = "bmatcuk/vagrant"
      version = "~> 4.0.0"
    }
  }
}

resource "vagrant_vm" "my_vagrant_vm" {
#  name = "vagrantbox"
  vagrantfile_dir = "."
  env = {
    BACKEND_IP = "192.168.50.10",
  }
  get_ports = true
}

output "host_port" {
  value = vagrant_vm.my_vagrant_vm.ports[0][0].host
}
```

En el fichero vemos además cómo podemos obtener el puerto en el que estará disponible el servicio supuestamente desplegado por la `Vagrantfile` como salida para otros despliegues. Además, vemos que la `Vagrantfile` estará en el mismo directorio que el fichero `.tf` (propiedad `vagrantfile_dir`), pero podemos poner la ruta que queramos. Si queréis probar con algo sencillo que no ocupe apenas recursos, podéis usar esta `Vagrantfile`, aunque puede usarse una del bloque de ejercicios de *Vagrant*.

```
Vagrant.configure("2") do |config|
  config.vm.box = "alpine/alpine64"
end
```

⚒️ N4TERRA_INFRALXD: Usar *Terraform* para desplegar una infraestructura de contenedores *LXD* *(0,2 puntos)*

🕵️ Descripción de la actividad

Consiste en aprender a **crear una infraestructura de contenedores LXD mediante *Terraform***, simulando así el despliegue de infraestructuras más complejas a un coste mínimo.

(NOTA: Este ejercicio requiere haber hecho al menos el Nivel 1 del bloque de contenedores de aplicaciones y el Nivel 2 del bloque de contenedores de SO)

🎖️ Resultados esperados

Eres capaz de crear una infraestructura de dos contenedores *LXD* comunicados pro una misma red local usando perfiles y probar que la conexión está operativa.

Información necesaria para su realización

Este ejercicio es una ampliación del que hicimos en el Nivel 3 sobre añadir una tarjeta de red a un contenedor *LXD*. Consiste en **crear otro contenedor *LXD Ubuntu* distinto,** y **hacer que use el mismo perfil** definido en dicho ejercicio para que también tenga una tarjeta de red interna `eth1`. Hecho esto, entrar en sesión en este segundo contenedor, darle una IP distinta a la del primero en su misma red, y **comprobar que ambos son capaces de verse en la red**.

Nivel 5 (Almirante): Interacción Terraform - Kubernetes

N5TERRA_K8S: Despliegue de un contenedor en *Kubernetes* con *Terraform* *(0,75 puntos)*

Descripción de la actividad

Consiste en **desplegar una infraestructura de contenedores *Docker*** sobre una instalación existente de ***Minikube*** gracias a *Terraform*.

(NOTA: Este ejercicio requiere haber hecho al menos el Nivel 1 del bloque de contenedores de aplicaciones y el Nivel 2 del bloque de Kubernetes)

Resultados esperados

Eres capaz de desplegar con *Terraform* un *deployment* sencillo sobre una instalación local de `minikube` en una *namespace* distinto, y comprobar que arranca correctamente.

Información necesaria para su realización

Para finalizar este boletín de ejercicios, vamos a usar *Terraform* para hacer un **despliegue de contenedores en un clúster *Kubernetes*** como el `minikube` local que instalamos en el bloque de ejercicios correspondiente. Una vez tenemos el `minikube` operativo, el *provider* `hashicorp/kubernetes` nos da los *resource types* y *data sources* necesarios para modelar los conceptos de *Kubernetes* que se ven en el bloque correspondiente.

Lo primero que debemos hacer es inicializarlo con una configuración como esta para que detecte el `minikube` local y se pueda comunicar con él:

```
terraform {
  required_providers {
    kubernetes = {
      source = "hashicorp/kubernetes"
      version = "2.11.0"
    }
  }
}

provider "kubernetes" {
  config_path    = "~/.kube/config"
  config_context = "minikube"
}
```

Y luego ya podremos usar los *resource types* que queramos para modelar el despliegue. Esta actividad consiste en desplegar como se hace habitualmente en *Terraform* esta configuración, que además introduce algunos conceptos nuevos de *Kubernetes* que no se ven en teoría:

- **Usar un *namespace* diferente del de por defecto para tener un clúster virtual independiente**: En la teoría de *Kubernetes* hablamos de los *namespaces*, pero no usamos uno en las prácticas. Ahora tenemos esta oportunidad de forma muy sencilla gracias a *Terraform*, desplegando todo bajo el *namespace* `terra`.
- **Usar límites de CPU, memoria y *requests***: *Kubernetes* permite fijar límites de uso de determinados recursos como los mencionados a sus *pods*, para garantizar una determinada calidad de servicio. Esto es un aspecto que no vimos en el tema de *Kubernetes* para no complicar más la explicación, pero que ahora podemos usar muy fácilmente gracias al *provider* de Kubernetes.
- El resto de los conceptos del fichero, como *labels*, *deployments*, etc. son los que ya vimos en el bloque correspondiente.

La especificación de lo que queremos desplegar es la siguiente, creando un *deployment* con 2 réplicas de `nginx` limitando sus recursos:

```
resource "kubernetes_namespace" "terra" {
  metadata {
    name = "k8s-terra"
  }
}

resource "kubernetes_deployment" "nginxserver" {
  metadata {
    name = "terraform-nginx"
    labels = {
      app = "EjemploServidor"
    }
    namespace = "k8s-terra"
  }

  spec {
    replicas = 2
    selector {
      match_labels = {
        app = "EjemploServidor"
      }
    }

    template {
      metadata {
        labels = {
          app = "EjemploServidor"
        }
      }
      spec {
        container {
          image = "nginx:latest"
          name  = "ejemplonginx"

          resources {
```

```
            limits = {
              cpu    = "0.5"
              memory = "512Mi"
            }
            requests = {
              cpu    = "250m"
              memory = "50Mi"
            }
          }
        }
      }
    }
  }
```

Una vez desplegado, debemos probar que efectivamente el *deployment* reside en un *namespace* independiente del `default`, haciendo la prueba que se ve en esta imagen:

```
ssiuser@vagrant:~/Documents/terra_k8s$ sudo kubectl --namespace=k8s-terra get de
ployments
NAME              READY   UP-TO-DATE   AVAILABLE   AGE
terraform-nginx   2/2     2            2           73s
ssiuser@vagrant:~/Documents/terra_k8s$ sudo kubectl get deployments
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
nginx      1/1     1            1           21h
postgres   1/1     1            1           4h26m
webapp     1/1     1            1           4h14m
ssiuser@vagrant:~/Documents/terra_k8s$
```

Insignias y Autoevaluación

Esta asignatura hace uso de insignias. Esto quiere decir que, cuando vayas desbloqueando niveles de maestría en la tecnología, podrás obtener **una insignia pública** que luego podrás usar en tu currículum gracias al soporte de estas del campus virtual. Este enlace te enseña a usar esta característica para poder unirlas a tu currículum: https://docs.moodle.org/all/es/Insignias. Cada insignia pública **lleva adjuntos los textos de las habilidades que desbloquea** que puedes consultar en la siguiente tabla, para que cualquiera los vea si decides usarlas y sepa lo que has conseguido hacer, validado por mí. De esta manera, mi intención es ayudarte a aumentar tus posibilidades de contratación en un futuro, **exponiendo claramente lo que has podido hacer en esta asignatura** referente a cada tecnología sobre la que has decidido evaluarte.

Rango de Maestría en la Tecnología y Habilidades Demostradas		Insignia Pública Obtenida
Nivel 1: Cabo de *Terraform*		
	Puede hacer una instalación funcional de *Terraform*	
	Es capaz de desplegar vía *Terraform* y el *provider* adecuado una imagen del *Docker Hub* y también una imagen local personalizada que ha creado ***(2 ejercicios)***	
Nivel 2: Sargento de *Terraform*		
	Puede exponer un servicio desplegado vía *Terraform* al exterior	
	Sabe desplegar grafos de recursos y planes de aplicación antes de hacer un despliegue de *Terraform*	
	Puede desplegar un contenedor *LXD* con *Terraform*	
Nivel 3: Teniente de Navío de *Terraform*		
	Puede comunicar contenedores de aplicaciones desplegados con *Terraform* mediante una red privada	
	Puede añadir un volumen a un contenedor *LXD* gracias a *Terraform*	
	Puede añadir una tarjeta de red manejando varios perfiles a un contenedor *LXD* gracias a *Terraform*	
Nivel 4: Capitán de Navío *Terraform*		
	Puede desplegar una máquina virtual *Vagrant* con *Terraform*	
	Puede comunicar contenedores de SO desplegados con *Terraform* mediante una red privada	
Nivel 5: Almirante *Terraform*		
	Puede desplegar un contenedor en un clúster *Kubernetes* mediante *Terraform*	

	**Cruz con el distintivo blanco en *Terraform***: Ha completado con éxito **2,2 puntos** (aproximadamente el 80%) de las tareas correspondientes de los 5 niveles de la parte de *Terraform*, demostrando así su maestría en esta tecnología y estar preparado para usarla en tu futuro trabajo de forma efectiva.