IMPLEMENTACIÓN DE UN BALANCEADOR DE CARGA HAPROXY CON DOS SERVIDORES DE BACKEND EN MÁQUINAS VIRTUALES VAGRANT Y CONTENEDORES LXD

Pedroza Francisco Javier; Tobar Luis Felipe;
francisco.pedroza@uao.edu.co
luis.tobar@uao.edu.co
Facultad de Ingeniería, Departamento de Electrónica y Automática.
Universidad Autónoma de Occidente
Cali, Colombia

Abstract* — *in this project a load balancer will be implemented in a specific configuration. Three virtual machines should interact as servers where one of them contains a frontend with HAProxy for balance the backends calling that are in the other two servers. The backends have to attend a number of clients petitions depending on the HAProxy configuration and have backup servers for a situation in which the backends are down or without capacity. The Haproxy , the backends and the backups backends should be in LXD containers for their functioning and should be implemented with provisioning.

Keywords – balancer, virtual machine, frontend, backend, HAProxy, backup, LXD container, provisioning.

I. INTRODUCCIÓN

HaProxy es un software de código abierto (proxy de alta disponibilidad) hace de intermediario entre las conexiones de un cliente y un servidor de destino. es decir que el proxy recibe peticiones para acceder a una u otra página en internet. su uso más común es mejorar el rendimiento y la confiabilidad de un entorno de servidor distribuyendo la carga de trabajo entre múltiples servidores. En relación con el balanceo de carga, HAProxy utiliza una lista de control de acceso(ACL) utilizada para seleccionar servidores o bloquear solicitudes permitiendo un reenvío flexible del tráfico de red. Un backend entonces es un conjunto de servidores que reciben solicitudes re-enviadas definidas por el frontend y puede contener uno o varios servidores.
Un balanceo de carga de capa 4 es la forma más sencilla de balancear el tráfico de red a múltiples servidores. El balanceo de carga de esta manera re-enviará el tráfico del usuario según el rango de la IP y el puerto.

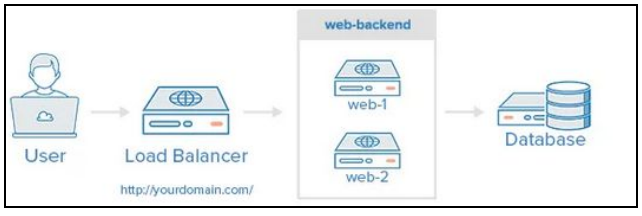

Figura 1: Diagrama de balanceador de carga de capa 4.

Al acceder al balanceador de carga este reenvía la solicitud al grupo de servidores de backend. En general, todos los servidores deben ofrecer contenido idéntico; de lo contrario, el usuario podría recibir contenido inconsistente. Uno de los algoritmos usado para balanceo de carga por defecto es el denominado “roundrobin” que selecciona por turno los servidores permitiendo así un equilibrio de carga de las solicitudes recibidas.

la instalación y configuración de HAProxy junto con los servidores de backend se encuentran contenidos dentro de máquinas virtuales. Los contenedores son útiles debido a que aíslan aplicaciones del resto del sistema. Son portátiles, fáciles de clonar y trasladar a otros sistemas operativos. LXD está diseñado para contener un sistema operativo completo. esto lo hace útil en diferentes escenarios por ejemplo y como es el caso instalar un servidor http y crear un sitio web en su interior. LXD no virtualia hardware lo que significa que es muy rápido, ofreciendo una velocidad de ejecución casi nativa.

II. PLANTEAMIENTO DEL PROBLEMA

El reto planteado como primer micro-proyecto de la asignatura de Computación en la nube consiste en implementar sobre tres máquinas virtuales (utilizando VirtualBox y Vagrant) un balanceador de carga utilizando HAProxy (High Availability Proxy). Sobre la primera máquina virtual se debe crear un contenedor LXC el cual tenga instalado HAProxy, que debe redireccionar las peticiones hechas por los clientes a dos servidores con Apache que deben correr sobre un contenedor LXC cada uno, contenedores los cuales a su vez deben estar cada uno en una de las dos máquinas virtuales restantes. A continuación se presenta una diagrama que permite entender gráficamente lo expresado anteriormente:

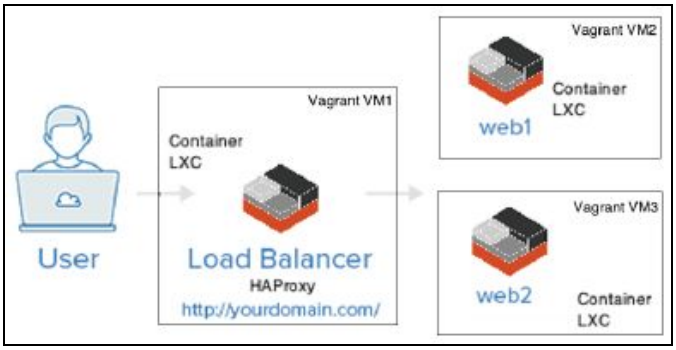

Figura 2: Diagrama explicativo del problema planteado para el primer micro-proyecto.

Con base en lo anterior se establecen los requerimientos generales que se deben cumplir al momento de presentar una solución: El primer requerimiento es que las peticiones realizadas por los clientes deben hacerse directamente al balanceador de carga y no a los servidores web, ya que debe ser a través de la configuración de este que se debe definir la forma en cómo se van a repartir las peticiones entre los dos servidores. El siguiente requerimiento es que los dos servidores web

nombrados anteriormente sólo deben tener apache ejecutándose, mientras que en el balanceador se ejecutará HAProxy. Como tercer requerimiento se hace referencia a la visualización del estado y estadísticas detalladas de los servidores web, lo que se debe hacer a través de la GUI del balanceador de carga que será accesible desde la máquina anfitriona. También se debe tener dos servidores de backup (en contenedores LXC) que entren en funcionamiento en caso de que los dos principales se caigan. Finalmente se deben hacer pruebas de cargas con JMeter para caracterizar la respuesta del sistema y además realizar aprovisionamiento para que todas las configuraciones que se hagan para cumplir con los anteriores requerimientos se realicen de forma automática desde que se crean las máquinas virtuales.

II. DESCRIPCIÓN DE LA SOLUCIÓN

El primer reto al que se le debe hacer frente es que tanto los contenedores que contienen los servidores web como el contenedor que tiene HAProxy estarán en máquinas virtuales diferentes, por lo que para que en la configuración del archivo haproxy.cfg se puedan escoger los otros servidores como parte del backend se sugiere utilizar Clustering LXD, ya que en "LXD se puede ejecutar en modo de agrupación en clúster, donde cualquier número de servidores LXD comparten la misma base de datos distribuida y se puede administrar de manera uniforme mediante el cliente LXC o la API REST."[1]

El reto que involucra el clustering surge al intentar aprovisionarse, ya que cuando se hace de forma manual sólo se necesita saber la clave para ingresar un nodo nuevo al clúster, pero para aprovisionarse se debe recurrir a un archivo de configuración tipo yaml. Para poder utilizar este método se recurre a la bandera ***preseed*** con la que cuenta LXD, y en la misma documentación se especifica los que debe tener dicho archivo tanto para el primer nodo con el que se crea el nodo como para los demás que se unan posteriormente :[1]

Para el primer nodo se debe establecer en el apartado de **config** la contraseña y la *https_address* (que será la misma IP con la que se creó la máquina virtual que tendrá dicho nodo) donde si no se especifica el puerto será por defecto el 8443. Luego se puede configurar la underlay subnetwork que se conoce con el nombre **lxdfan0**, y precisamente la IP que se configura aquí debe ser la de identificación que se usa para referirse a la subred completa, que sería cambiando el último número de las IP de las máquinas virtuales por cero. También se configura el **storage_pool**, con driver tipo dir y almacenamiento local y finalmente lo más importante es activar el uso de clustering dejando la variable *enabled* en true y estableciendo el *server_name*.

Para los siguientes nodos que se quieran adicionar al cluster se debe utilizar un yaml en el cual se especifica que se va a utilizar clustering, nuevamente con la variable *enabled* en true. Se debe especificar el nombre del nuevo nodo en *server_name,* junto con la IP que tendrá (en el caso de esta implementación será la misma que la máquina virtual donde se realiza el nuevo nodo) y la IP del primer nodo que se creó con el clúster. El *storage_pool* se configura de forma local y finalmente se debe adjuntar la contraseña con la cual se hizo el anterior nodo y el certificado que se genera al crear el clúster.

Ahora bien, debido a que el certificado se genera sólo en la primera máquina virtual donde se hace la configuración del primer nodo (que hará parte del clúster(se hace necesario enviar dicho certificado a las demás máquinas para que pueda incluirse en los archivos de configuración yaml y que se puedan unir al clúster los nuevos nodos. Para esto se plantea aprovechar la carpeta /vagrant que comparten las máquinas virtuales creadas con el mismo VagrantFile, ya que se puede copiar el archivo que contiene el certificado (llamado **server.crt**) a dicho directorio y ser visto usado por las demás máquinas virtuales. Una vez se tiene realizado el clustering con LXD es posible crear contenedores LXC desde cualquiera de las máquinas virtuales y poder verlas desde las demás, e incluso se pueden crear contenedores desde un nodo cualquiera e indicarle que la ubicación del contenedor sea en otro nodo del clúster, lo cual facilita la configuración y aprovisionamiento de los contenedores que tendrán los servidores web, los de backup y el del balanceador.

Con base en lo anterior se plantea un orden en el cual se realizará el aprovisionamiento, y es que para que desde cualquier nodo se puedan crear contenedores LXC visibles y accesibles por los otros se debe primero completar en su totalidad la configuración del clustering, que implica crear las tres máquinas virtuales y añadirlas como nodos al clúster, para luego crear los contenedores LXC en la última máquina que se añada al clúster. Otro orden que se debe respetar al hacer el aprovisionamiento es que primero se deben crear los contenedores con los servicios web y los de backup antes de configurar el que tendrá el HAProxy, ya que si no se han creado antes los que harán parte del backend pueden surgir el errores al habilitar el haproxy y detener el aprovisionamiento.

III. PRUEBAS

Para la solución del proyecto se crearon tres máquinas virtuales: la primera se le llamó vmBalanceador, donde se creó el primer nodo que haría parte del clúster LXD y que sería el mismo que contendría el contenedor LXC donde se ejecuta el HAProxy. A las otras dos máquinas virtuales se les llamó vmServidor1 y vmServidor2, las cuales son las que tendrían los contenedores de los servidores web apache y los de backup. A continuación se muestra un diagrama que ejemplifica lo descrito anteriormente:

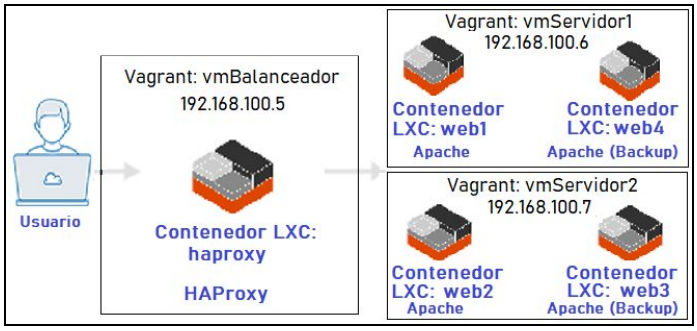

Figura 3: Diagrama explicativo de estructura de máquinas virtuales y contenedores LXC implementada.

A partir de la solución planteada se iniciaron con las pruebas de clustering, que en primera instancia se hicieron manuales con el comando **lxd init** para la creación del primer nodo y también para los otros dos desde máquinas virtuales distintas. Estas pruebas se hicieron basadas en los videos del canal "Just me and Opensource" de youtube, lo que permitió ver que se debía configurar la underlay subnetwork (lxdfan0) por la IP de identificación de las máquinas virtuales. En el video se hace de forma manual. modificando por medio de vim el

[1] Clustering. LXC system container manager. Disponible en https://lxd.readthedocs.io/en/latest/clustering/

lxdfan0 con el comando '**lxc network edit lxdfan0**', pero al leer la documentación de los archivos yaml para utilizar el **preseed** se pudo ver que dicha configuración se podía modificar durante la configuración y no necesariamente después.

Una vez se entendió los parámetros que se debían configurar para realizar correctamente el clustering se empezó a hacer pruebas para enviar el certificado que se genera hacía la carpeta vagrant que comparten las tres máquinas virtuales. Debido a que el comando utilizado para instalar lxd fue '**sudo snap install lxd**', la carpeta donde se crea el archivo *server.crt* (el cual contiene el certificado) está en la ruta '*/var/snap/lxd/common/lxd/server.crt*', y con el comando sudo cp se copió dicho archivo a la ruta '*/vagrant*'.

Luego de ésto se procedió a automatizar dicho proceso a través de un archivo de aprovisionamiento shell llamado "clusterBalanceador.sh", que realiza el clúster, el primer nodo y la copia del certificado en la carpeta vagrant una vez se crea la máquina virtual vmBalanceador:

```
#Balanceador
config.vm.define :vmBalanceador do |vmBalanceador|
vmBalanceador.vm.box = "bento/ubuntu-20.04"
vmBalanceador.vm.network :private_network, ip: "192.168.100.5"
vmBalanceador.vm.provision "shell", path: "clusterBalanceador.sh"
vmBalanceador.vm.hostname = "vmBalanceador"
vmBalanceador.vm.provider "virtualbox" do |v|
   v.name = "vmBalanceador"
   v.memory =1024
   v.cpus=1
end
end
```

Figura 4: Extracto de código del Vagrantfile utilizado para crear la máquina virtual vmBalanceador y aprovisionar a partir del archivo clusterBalanceador.sh.

Para poder aprovisionar la creación del cluster se crea un archivo yaml previamente configurado con todos los parámetros necesarios (como se explicó en el apartado de la descripción de la solución), el cual se le adiciona al comando **lxc init --preseed** por medio del comando '*cat <<EOF*", "este tipo de redirección ordena al shell que lea la entrada de la fuente actual hasta que se vea una línea que contenga solo la palabra (sin espacios en blanco al final). Todas las líneas leídas hasta ese punto se usan luego como entrada estándar para un comando"[2]:

```
cat <<EOF | lxd init --preseed
config:
  core.https_address: 192.168.100.5:8443
  core.trust_password: admin
networks:
- config:
    bridge.mode: fan
    fan.underlay_subnet: 192.168.100.0/24
  description: ""
  name: lxdfan0
  type: ""
  project: default
storage_pools:
- config: {}
  description: ""
  name: local
  driver: dir
profiles:
- config: {}
  description: ""
  devices:
    eth0:
      name: eth0
      network: lxdfan0
      type: nic
    root:
      path: /
      pool: local
      type: disk
  name: default
cluster:
  server_name: vmBalanceador
  enabled: true
  member_config: []
  cluster_address: ""
  cluster_certificate: ""
  server_address: ""
  cluster_password: ""
EOF
```

Figura 5: Configuración del primer nodo para crear el clúster.

Una vez se ha creado el clúster y el primer nodo ya hace parte de él se realiza el proceso de copiarlo desde la ruta donde se crea hasta la carpeta /vagrant (se deja también una copia en la ruta /home/vagrant para poder visualizar fácilmente el contenido del certificado):

```
sudo cp /var/snap/lxd/common/lxd/server.crt /home/vagrant

sudo cp /home/vagrant/server.crt /vagrant
```

Figura 6: Extracto de código del archivo clusterBalanceador.sh para aprovisionar el envió del certificado a la carpeta compartida '/vagrant' por las tres máquinas virtuales.

Lo siguiente que se realiza al tener creado el clúster son las otras dos máquinas virtuales, que por medio de archivos de aprovisionamiento shell (que se les nombró "clusterServidor1.sh" y "clusterServidor2.sh") se les incorpora al clúster creado anteriormente con el certificado que se envió a la rita /vagrant/server.crt:

[2] ¿Cómo funciona "cat << EOF" en bash? [En línea]. Disponible en https://www.dokry.com/5862

```
#Servidor 1
 config.vm.define :vmServidor1 do |vmServidor1|
 vmServidor1.vm.box = "bento/ubuntu-20.04"
 vmServidor1.vm.network :private_network, ip: "192.168.100.6"
 vmServidor1.vm.provision "shell", path: "clusterServidor1.sh"
 vmServidor1.vm.hostname = "vmServidor1"
 vmServidor1.vm.provider "virtualbox" do |v|
    v.name = "vmServidor1"
    v.memory =1024
    v.cpus=1
 end
 end

#Servidor 2
 config.vm.define :vmServidor2 do |vmServidor2|
 vmServidor2.vm.box = "bento/ubuntu-20.04"
 vmServidor2.vm.network :private_network, ip: "192.168.100.7"
 vmServidor2.vm.provision "shell", path: "clusterServidor2.sh"
```

Figura 7: Extracto de código del Vagrantfile utilizado para crear las máquinas virtuales vmServidor1 y vmServidor2, y aprovisionarlas a partir de los archivos clusterServidor1.sh y clusterServidor2.sh (la máquina virtual vmServidor2 tiene más archivos de configuración que no se muestran en la figura).

Ahora bien, el contenido de cada uno de los archivos de aprovisionamiento shell para cada máquina virtual tiene los comandos **sudo snap install lxd** y **sudo gpasswd -a vagrant lxd** al inicio, para posteriormente realizar el aprovisionamiento de añadir sus respectivos nodos al clúster con **lxd init --preseed**. Para ésto se trae primero el archivo server.crt que tiene el certificado a la ruta /home/vagrant de cada máquina para verlo, el cual tiene el siguiente formato:

```
-----BEGIN CERTIFICATE-----
MIICDzCCAZagAwIBAgIRAOzfxzKBfV9FT8yDxMOASqkwCgYIKoZIzj0EAwMwODEc
MBoGA1UEChMTbGludXhjb250YWluZXJzLm9yZzEYMBYGA1UEAwwPcm9vdEBwcnVl
YmFjbHVzMB4XDTIwMTAyODE4MjM0M1oXDTMwMTAyNjE4MjM0M1owODEcMBoGA1UE
ChMTbGludXhjb250YWluZXJzLm9yZzEYMBYGA1UEAwwPcm9vdEBwcnVlYmFjbHVz
MHYwEAYHKoZIzj0CAQYFK4EEACIDYgAEHehLHciRAgrRKk6Zkgm0aSX1RxrWBRKd
s8AKojAPCEJcXRPbeS7eItpUDp1U42JfgTSVaqj4y0je5b2miUJC7DEhBDqxGUre
V39V+MxqlNH+oAHI5XXc0tlhiS62wMmdo2QwYjAOBgNVHQ8BAf8EBAMCBaAwEwYD
VR0lBAwwCgYIKwYBBQUHAwEwDAYDVR0TAQH/BAIwADAtBgNVHREEJjAkggpwcnVl
YmFjbHVzhwR/AAABhxAAAAAAAAAAAAAAAAAAAAABMAoGCCqGSM49AMDA2cAMGQC
MHCt1E44Zn7Gc2uJVUYGFSIrMX4a7u3uRxpt0raocuVQJEf2sMRTJBhZLyUu8tjz
7wIwaJU+EfMpLRsSAQibfQoU3bK4EWuNGomGcPtmP6AB6C8icfzLZqVMDaIrVWop
l88i
-----END CERTIFICATE-----
```

Figura 8: Formato de certificados generados al realizar el clúster LXD en el archivo sever.crt.

Como se puede ver en la anterior figura este certificado cuenta con 14 líneas que tienen un extenso código que se necesita para poder adicionar un nuevo nodo al clúster. Debido a que realizaron previamente las pruebas manuales a partir de los videos de "Just me and Opensource" se pudo comprobar que siempre los códigos tienen dicha cantidad de líneas, lo que es un dato a recordar para siguientes etapas del aprovisionamiento que se comentarán más adelante.

El siguiente paso en el aprovisionamiento de las máquinas vmServidor1 y vmServidor2 fué crear un archivo yaml con todas las configuraciones necesarias para añadir el nuevo nodo al clúster ya existente. En este caso no se utilizó el comando cat <<EOF antecediendo al **lxd init --preseed**, sino que se usó cat <<TEST> para primero crear los archivos *cluster1cfg.yaml* y *cluster2cfg.yaml*, que tienen todos los parámetros predefinidos para añadir los nuevos nodos desde las máquinas virtuales vmServidor1 y vmServidor2 respectivamente:

```
touch /home/vagrant/cluster2cfg.yaml
cat <<TEST> /home/vagrant/cluster2cfg.yaml
config: {}
networks: []
storage_pools: []
profiles: []
cluster:
  server_name: vmServidor2
  enabled: true
  member_config:
  - entity: storage-pool
    name: local
    key: source
    value: ""
    description: '"source" property for storage pool "local"'
  cluster_address: 192.168.100.5:8443
  cluster_certificate: |
  server_address: 192.168.100.7:8443
  cluster_password: admin
TEST
```

Figura 9: Ejemplo de archivo cluster2cfg.yaml para añadir un nuevo nodo al clúster existente, ya sea desde las máquinas virtuales vmServidor1 o vmServidor2.

Como se observa en la anterior figura, en la línea 15 del archivo que se crea (cluster2cfg.yaml) hay una parámetro llamada *cluster_certificate*, que es donde se debe adicionar el certificado mostrado en la figura 8. Para lo anterior se debe tener en cuenta que el certificado deberá insertarse entre las líneas 15 y 16 que hay actualmente en el archivo yaml, y que debe estar correctamente identado. Para lo anterior se procede primero a darle una indentación de 4 espacios a todas las líneas del archivo server.crt con el comando sed, y guardar dicho cambio en un nuevo archivo llamado serverm1.crt, para luego nuevamente con el comando sed indicar que se desea insertar luego de la línea 15 del archivo cluster2.cfg.yaml el contenido desde la línea 1 a la 14 del archivo serverm1.crt, así:

```
sed 's/^/    /g' server.crt > serverm1.crt

sed -i 15r<(sed '1,14!d' serverm1.crt) cluster2cfg.yaml
```

Figura 10: Comandos utilizados para insertar correctamente el certificado del archivo server.crt en el archivo de configuración cluster2cfg.yaml.

Finalmente para terminar de añadir los nuevos nodos al clúster, tanto para vmServidor1 y vmServidor2, se utiliza los archivos cluster1cfg.yaml y cluster2cfg.yaml modificados (con el certificado en su contenido) para ejecutar **'cat cluster1cfg.yaml | lxd init --preseed**' y '**cat cluster2cfg.yaml | lxd init --preseed**' respectivamente.

El siguiente paso a configurar es el proceso de aprovisionamiento de la creación de los contenedores LXC, el cual se debe realizar luego de la configuración previa del clustering con LXD entre las máquinas virtuales. Como se explicó en el apartado de descripción de la solución, se deben crear primero los contendores que tendrán apache, los cuales son web1, web2 y los de backup web3 y web4, para finalmente crear el contenedor que tendrá HAProxy. Este orden es el que se sigue para evitar que el proceso de aprovisionamiento se detenga, ya que si se hace primero la configuración del archivo haproxy.cfg sin tener creados los servidores que harán parte del backend, el servicio de haproxy no se iniciará correctamente. Con base en lo anterior se utilizan 3 archivos de aprovionamiento (vmServidor1_cfg.sh, vmServidor2_cfg.sh y vmBalanceador_cfg.sh):

```
#Servidor 2
 config.vm.define :vmServidor2 do |vmServidor2|
 vmServidor2.vm.box = "bento/ubuntu-20.04"
 vmServidor2.vm.network :private_network, ip: "192.168.100.7"
 vmServidor2.vm.provision "shell", path: "clusterServidor2.sh"
 vmServidor2.vm.provision "shell", path: "vmServidor1_cfg.sh"
 vmServidor2.vm.provision "shell", path: "vmServidor2_cfg.sh"
 vmServidor2.vm.provision "shell", path: "vmBalanceador_cfg.sh"
 vmServidor2.vm.hostname = "vmServidor2"
 vmServidor2.vm.provider "virtualbox" do |v|
    v.name = "vmServidor2"
    v.memory =1024
    v.cpus=1
 end
 end
```

Figura 11: Extracto de código del Vagrantfile utilizado para aprovisionar la creación los contenedores LXC que tendrán apache y el que tendrá HAProxy.

Como se puede ver en la anterior figura todos los archivos de aprovisionamiento shell para crear todos los contenedores LXC están en la máquina virtual vmServidor2, y ésto es gracias al clustering realizado, ya que se puede especificar al crear cada contenedor cuál es el nodo target en el cual se quiere crear. Es así como se realiza la creación de los contenedores LXC que contienen los servidores principales de backend web1 y web2, los servidores de backup web3 y web4 y el de HAProxy:

```
echo '----CREANDO CONTENEDOR WEB1----'
lxc launch ubuntu:18.04 web1 --target vmServidor1
sleep 5
echo '----UPDATE AND UPGRADE EN EL CONTENEDOR----'
lxc exec web1 -- sudo apt update
lxc exec web1 -- sudo apt upgrade -y
sleep 10
echo '----INSTALANDO APACHE----'
lxc exec web1 -- apt-get install apache2 -y
echo '----HABILITANDO APACHE----'
lxc exec web1 -- systemctl enable apache2
sleep 10

echo "Configurando index.html"

touch /home/vagrant/index.html
```

Figura 12: Ejemplo de comandos de aprovisionamiento para crear un contenedor LXC con servicio apache.

En la figura anterior se visualiza como a pesar de que el archivo shell donde se está creando el contendor web1 hace parte del aprovisionamiento de vmServidor2. al agregar la bandera **--target vmServidor1** es posible crearlo sobre dicho nodo del cluster, y es así como se pueden definir comandos para ese contenedor desde cualquier otro nodo, que para el caso de los servidores web son comandos de actualización, la instalación de apache2 y la habilitación del servicio, para posteriormente crearles un archivo index.html con un diseño propio que es el que se muestra por medio de la red local de la máquina anfitriona. Para poder establecer dicho archivo index.html como el que debe usar cada servidor web apache se debe guardar en el directorio */var/ww/html/index.html* de cada uno de los contenedores, y para que se detecte los cambios se debe reiniciar el servicio de apache, así:

```
lxc file push /home/vagrant/index.html web1/var/www/html/index.html
echo '----RESTART A APACHE----'
lxc exec web1 -- systemctl restart apache2
```

Figura 13: Ejemplo de envío del archivo index. html al contenedor web1 y reinicio de apache para detectar el cambio.

El el mismo archivo shell anterior se hace la creación del servidor de back up web3 (en vmServidor1_cfg.sh) y la c creación de los contenedores web 2 y web4 se hace de la misma manera pero en el archivo de configuración vmServidor2_cfg.sh (cambiando el contenido de los index.html).

El último contenedor que se crea es el de HAProxy, el cual se aprovisiona en el archivo vmBalanceador_cfg.sh. En éste se sigue un proceso muy parecido al que se hace con los otros contendores, sólo que en lugar de instalar apache2 se realiza la instalación de haproxy y además el archivo que crea para éste no es un html, sino el archivo de configuración haproxy donde se especifica el modo en el que va a funcionar el balanceador de carga.

```
echo '----CREANDO CONTENEDOR HAPROXY----'
lxc launch ubuntu:18.04 haproxy --target vmBalanceador
sleep 5
echo '----UPDATE AND UPGRADE EN EL CONTENEDOR----'
lxc exec haproxy -- sudo apt update
lxc exec haproxy -- sudo apt upgrade -y
sleep 10
echo '----INSTALANDO HAPROXY----'
lxc exec haproxy -- apt install haproxy -y
echo '----HABILITANDO HAPROXY----'
lxc exec haproxy -- systemctl enable haproxy
sleep 5

echo "----Configurando el haproxy.cfg con cat----"
touch /home/vagrant/haproxy.cfg
```

Figura 14: Ejemplo de comandos de aprovisionamiento para crear un contenedor LXC con servicio HAProxy.

```
defaults
    log global
    mode    http
    option  httplog
    option  dontlognull
        timeout connect 5000
        timeout client  50000
        timeout server  50000
    errorfile 400 /etc/haproxy/errors/400.http
    errorfile 403 /etc/haproxy/errors/403.http
    errorfile 408 /etc/haproxy/errors/408.http
    errorfile 500 /etc/haproxy/errors/500.http
    errorfile 502 /etc/haproxy/errors/502.http
    errorfile 503 /etc/haproxy/errors/503.http
    errorfile 504 /etc/haproxy/errors/504.http


backend web-backend
   balance roundrobin
   stats enable
   stats auth admin:admin
   stats uri /haproxy?stats
   stats refresh 10s

   option allbackups
   server web1 web1.lxd:80 check
   server web2 web2.lxd:80 check
   server web3 web3.lxd:80 check backup
   server web4 web4.lxd:80 check backup

frontend http
  bind *:80
  default_backend web-backend
```

Figura 15: Archivo de configuración *haproxy.cfg* creado en el aprovisionamiento.

Como se ve en la figura 15 se resalta cuál es la ruta donde se debe enviar el archivo html personalizado en caso de que se presente el error de que no hay ningún servidor disponible para atender las peticiones.

También se especifica el algoritmo que se utilizará para repartir los requerimientos hacía los servidores, que en este caso será **roundrobin** (que alterna de forma equitativa las peticiones). También se habilita la visualización de las estadísticas por medio de añadir a la URL la extensión */haproxy?stats,* se establece un usuario y contraseña para poder visualizarlo. el balanceador por defecto y se añade que se actualicen las estadísticas de forma automática cada 10 segundos. Finalmente en este archivo se habilita una de las formas de utilizar servidores de backup con *option allbackups,* se definen los servidores web1 y web2 como los principales, y debido a que al aprovisionar no se saben las direcciones IP con las que se crearan los contenedores se recurre a que LXD permite referirse a dichas IP añadiendo **.lxd** luego del nombre del servidor eso hace referencia a los nombres de dominio y sus correspondientes IP (DNS) de cada uno de los contenedores. Seguido a la IP se agrega para todos los servidores *check,* para poder recibir health bits, relacionados al monitoreo de la "salud" de cada servidor, indicando su correcto funcionamiento. Además para los servidores web3 y web4 que serán de backup se les agrega **backup** luego del health checking, lo que permite al balanceador identificarlos para backup.

Finalmente se envía el archivo de configuración modificado a la ruta */etc/haproxy/haproxy.cfg* del contenedor haproxy, y se reinicia el servicio de HAProxy para que se apliquen los cambios, así:

```
lxc file push /home/vagrant/haproxy.cfg haproxy/etc/haproxy/haproxy.cfg
echo '----RESTART A HAPROXY----'
lxc exec haproxy -- systemctl restart haproxy
```

Figura 16: Envío del archivo haproxy.cfg al contenedor harpoxy y reinicio del servicio HAProxy para detectar el cambio.

IV.RESULTADOS

A continuación se muestra los nodos que se crean al realizar el proceso de clustering desde la máquina virtual vmBlanceador:

```
vagrant@vmServidor1: $ lxc cluster list
+-------------+---------------------------+----------+--------+-----------------+--------------+----------------+
|    NAME     |            URL            | DATABASE | STATE  |     MESSAGE     | ARCHITECTURE | FAILURE DOMAIN |
+-------------+---------------------------+----------+--------+-----------------+--------------+----------------+
| vmBalanceador | https://192.168.100.5:8443 | YES      | ONLINE | fully operational | x86_64       | default        |
+-------------+---------------------------+----------+--------+-----------------+--------------+----------------+
| vmServidor1 | https://192.168.100.6:8443 | YES      | ONLINE | fully operational | x86_64       | default        |
+-------------+---------------------------+----------+--------+-----------------+--------------+----------------+
| vmServidor2 | https://192.168.100.7:8443 | YES      | ONLINE | fully operational | x86_64       | default        |
+-------------+---------------------------+----------+--------+-----------------+--------------+----------------+
```

Figura 17: Lista de nodos añadidos al cluster LXD.

Posterior a ello se comprueba la creación de los contenedores LXC desde la máquina virtual vmServidor1, evidenciando que a pesar de que algunos contenedores estén sobre otros nodos son visibles desde cualquier otro:

```
vagrant@vmServidor1: $ lxc list
+---------+---------+--------------------+------+-----------+-----------+--------------+
|  NAME   |  STATE  |        IPV4        | IPV6 |   TYPE    | SNAPSHOTS |   LOCATION   |
+---------+---------+--------------------+------+-----------+-----------+--------------+
| haproxy | RUNNING | 240.5.0.100 (eth0) |      | CONTAINER | 0         | vmBalanceador |
+---------+---------+--------------------+------+-----------+-----------+--------------+
| web1    | RUNNING | 240.6.0.184 (eth0) |      | CONTAINER | 0         | vmServidor1  |
+---------+---------+--------------------+------+-----------+-----------+--------------+
| web2    | RUNNING | 240.7.0.240 (eth0) |      | CONTAINER | 0         | vmServidor2  |
+---------+---------+--------------------+------+-----------+-----------+--------------+
| web3    | RUNNING | 240.7.0.103 (eth0) |      | CONTAINER | 0         | vmServidor2  |
+---------+---------+--------------------+------+-----------+-----------+--------------+
| web4    | RUNNING | 240.6.0.19 (eth0)  |      | CONTAINER | 0         | vmServidor1  |
+---------+---------+--------------------+------+-----------+-----------+--------------+
```

Figura 18: Lista de contenedores LXC creados con el aprovisionamiento.

Ahora bien, mediante el comando curl apuntado a la ip del contenedor que contiene HAProxy se puede verificar el buen funcionamiento del balanceador de carga, alternando las peticiones recibidas a los servidores de backend principales (estando ambos activos):

```
vagrant@vmServidor1: $ curl 240.5.0.100
<!DOCTYPE html>
<html>
<head>
<title>Pagina de prueba</title>
</head>
<body background="https://www.practia.global/Industrias/PublishingImages/BannerTecnologia.jpg">
<h1><center><p style="color:rgb(255,255,255);">Pagina web de prueba</p></center></h1>
<font size="3" face="Comic Sans MS, Arial, MS Sans Serif">
<center><p style="color:rgb(255,255,255);">Usted ahora mismo esta usando el servidor web1</p></center></font>
<h2><center><p style="color:rgb(255,255,255);">--usando web1--</p><center></h2>
</body>
</html>
vagrant@vmServidor1: $ curl 240.5.0.100
<!DOCTYPE html>
<html>
<head>
<title>Pagina de prueba</title>
</head>
<body background="https://www.practia.global/Industrias/PublishingImages/BannerTecnologia.jpg">
<h1><center><p style="color:rgb(255,255,255);">Pagina web de prueba</p></center></h1>
<font size="3" face="Comic Sans MS, Arial, MS Sans Serif">
<center><p style="color:rgb(255,255,255);">Usted ahora mismo esta usando el servidor web2</p></center></font>
<h2><center><p style="color:rgb(255,255,255);">--usando web2--</p><center></h2>
</body>
</html>
vagrant@vmServidor1: $
```

Figura 19: Verificación del funcionamiento del balanceador de carga mediante comandos en terminal**.**

También se verifica el correcto funcionamiento del balanceador desde la red local de la máquina anfitriona,que es posible se hace posible gracias a la creación y configuración de un dispositivo para redireccionamiento de puertos, que para este caso se usó el puerto 1080. Esto permite a un equipo externo el acceso mediante una dirección privada dentro de una misma red LAN O WLAN.

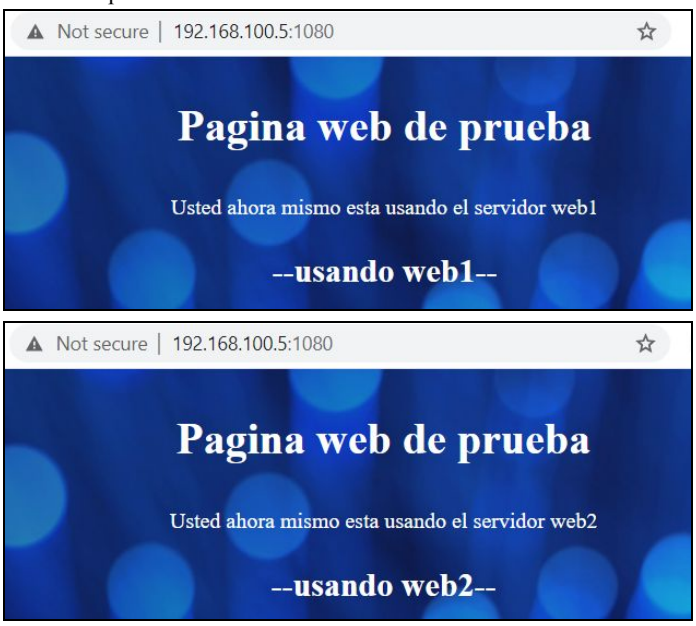

Figura 20: Verificación de funcionamiento del balanceador de carga mediante el navegador web**.**

Ahora se prueban los servidores de backup, para verificar la configuración del balanceador de carga que se hizo, donde al momento de que los servidores principales no estén disponibles, los servidores de backup deben entran en funcionamiento:

```
vagrant@vmServidor1: $ lxc list
+---------+---------+--------------------+------+-----------+-----------+--------------+
|  NAME   |  STATE  |        IPV4        | IPV6 |   TYPE    | SNAPSHOTS |   LOCATION   |
+---------+---------+--------------------+------+-----------+-----------+--------------+
| haproxy | RUNNING | 240.5.0.100 (eth0) |      | CONTAINER | 0         | vmBalanceador |
+---------+---------+--------------------+------+-----------+-----------+--------------+
| web1    | STOPPED |                    |      | CONTAINER | 0         | vmServidor1  |
+---------+---------+--------------------+------+-----------+-----------+--------------+
| web2    | STOPPED |                    |      | CONTAINER | 0         | vmServidor2  |
+---------+---------+--------------------+------+-----------+-----------+--------------+
| web3    | RUNNING | 240.7.0.103 (eth0) |      | CONTAINER | 0         | vmServidor2  |
+---------+---------+--------------------+------+-----------+-----------+--------------+
| web4    | RUNNING | 240.6.0.19 (eth0)  |      | CONTAINER | 0         | vmServidor1  |
+---------+---------+--------------------+------+-----------+-----------+--------------+
vagrant@vmServidor1: $ curl 240.5.0.100
<!DOCTYPE html>
<html>
<head>
<title>Pagina de prueba</title>
</head>
<body background="https://www.practia.global/Industrias/PublishingImages/BannerTecnologia.jpg">
<h1><center><p style="color:rgb(255,255,255);">Pagina web de prueba</p></center></h1>
<font size="3" face="Comic Sans MS, Arial, MS Sans Serif">
<center><p style="color:rgb(255,255,255);">Usted ahora mismo esta usando el servidor web1</p></center></font>
<h2><center><p style="color:rgb(255,255,255);">--usando backup web3--</p><center></h2>
</body>
</html>
vagrant@vmServidor1: $ curl 240.5.0.100
<!DOCTYPE html>
<html>
<head>
<title>Pagina de prueba</title>
</head>
<body background="https://www.practia.global/Industrias/PublishingImages/BannerTecnologia.jpg">
<h1><center><p style="color:rgb(255,255,255);">Pagina web de prueba</p></center></h1>
<font size="3" face="Comic Sans MS, Arial, MS Sans Serif">
<center><p style="color:rgb(255,255,255);">Usted ahora mismo esta usando el servidor web2</p></center></font>
<h2><center><p style="color:rgb(255,255,255);">--usando backup web4--</p><center></h2>
</body>
</html>
```

Figura 21: Verificación de funcionamiento del balanceador de carga cuando los servidores principales están inactivos mediante comandos en terminal**.**

Figura 22: Verificación de funcionamiento del balanceador de carga cuando los servidores principales están inactivos mediante el navegador web**.**

Como se puede ver en las anteriores figuras los servidores de backup funcionan en alternancia acorde a la configuración del balanceador de carga. Cabe resaltar que en este tipo de configuración si uno de los dos servidores principales no está disponible el segundo estará recibiendo las peticiones sólo, y los servidores de backup solo funcionan en el momento en que ambos servidores principales no están disponibles. Igualmente en caso de que sólo estén funcionando los servidores de backup y uno de los dos deje de estar disponible, funcionará solamente el otro.También se aclara que los servidores de backup funcionarán hasta que los servidores principales vuelvan a estar disponibles (en ese momento toman prioridad y relevan automáticamente la función acorde a la configuración principal).

Ahora bien, como en la creación del archivo de configuración de HAproxy quedó activa la visualización de estadísticas de secciones para hacer pruebas de campo, se prueba esto ingresando desde el navegador a "http://192.168.100.5:1080/haproxy?stats" y escribiendo el usuario y contraseña que se estableció (admin y admin).

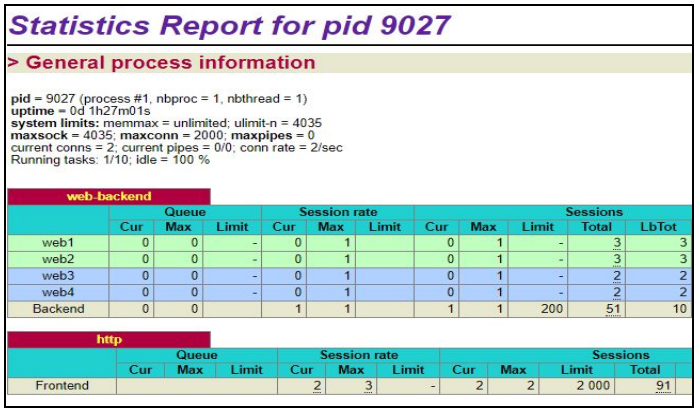

Figura 23: Interfaz de estadísticas de HAProxy.

Mediante la interfaz de estadísticas se puede evidenciar datos como la cantidad de peticiones que se están haciendo en un determinado lapso de tiempo, si estas peticiones fueron recibidas y si hubo respuesta, o si por el contrario fueron denegadas.También se puede observar que se tiene un límite por defecto para el backend, esto quiere decir que cuando alguno de los servidores supera el límite no queda funcional hasta bajar la carga, por eso la importancia de tener servidores de backup en un sitio web de alta afluencia.

Mediante una herramienta de Apache llamada JMeter (la cual permite medir la carga para analizar y medir el rendimiento de un servicio web) se hace una prueba donde se generan 4000 peticiones (de usuarios) en 1 segundo, repitiendo 1 sola vez la misma acción para un total de 8000 peticiones:

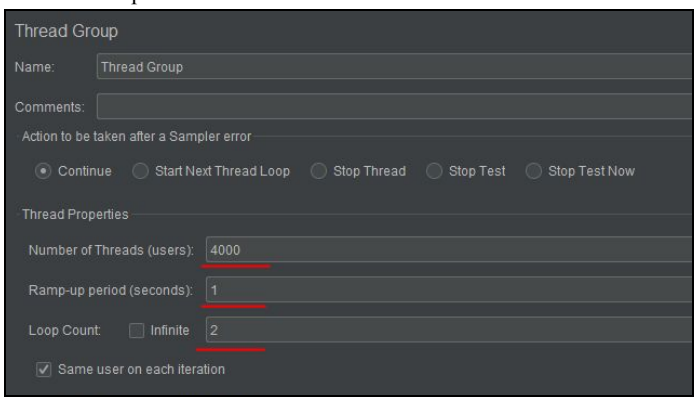

Figura 24: Interfaz de de JMeter para configurar pruebas de carga.

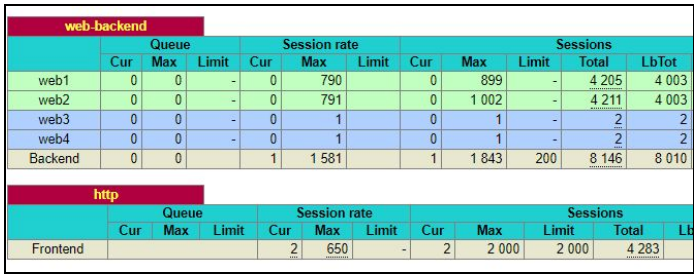

Figura 25: Interfaz de estadísticas HAProxy luego de pruebas de carga con JMeter.

Se observa que posterior a la prueba de carga web1 y web2 obtienen un número de peticiones en la cual al llegar a un límite en la primera y segunda intervención de ciclo de carga activan los servidores de backup en 1 ocasión , el total de peticiones al backend fue de 8146 rebosando lo solicitado desde JMeter y permitiendo generar una corta saturación a los servidores.

V. CONCLUSIONES

- El hecho de tener todos los contenedores LXC sobre una misma máquina hace poco robusto el servicio ante posibles errores, ya que si esa máquina falla todos los contenedores (los que tienen el servicios web apache, los de backup y el de HAProxy) dejan de funcionar. Es por esto que el clustering LXD es una manera muy efectiva de comunicar los contenedores LXC desde distintas máquinas virtuales, ya que le proporciona robustez al servicio, permitiendo que se puedan gestionar los contendores desde cualquier nodo que esté incluido en el clúster y que en caso de que se caiga una de las máquinas virtuales, el servicio prestado se sigue suministrando ya que sólo se caen los contenedores (con los servidores web) que hagan parte de dicha máquina.
- Dependiendo de la cualidades funcionales que se requieran para una implementación web donde se hagan peticiones a un backend el cual corre uno o varios procesos, se puede definir la cantidad de servidores implementar para evitar la caída de nuestro servicio y hacer una configuración de balanceo de carga eficiente permitiendo el mayor rendimiento posible.
- Mediante el uso de herramientas como JMeter se puede medir y analizar el comportamiento de los desarrollos buscando encontrar los valores límite de peticiones que puede soportar el servicio al cual se le implementó el balanceador de carga. Esto asumiendo una cantidad de peticiones posible proyectas a un ambiente de producción del desarrollo, y de esta manera es mucho más eficiente definir las características y servidores a implementar.

REFERENCIAS

[1] Binaria 2014.[en línea]. Disponible en https://www.blog.binaria.uno/2019/04/14/introduccion-a-haproxy-y-balanceadores-de-carga/. [Último acceso: 2020].

[2] Vidatecno 2019 [en línea]. Available: https://vidatecno.net/introduccion-a-los-contenedores-lxd-de-ubuntu/#. [Último acceso: 2020].

[3] HAPROXY 2019 [en línea]. Disponible en https://www.haproxy.com/blog/exploring-the-haproxy-stats-page/#:~:text=HAProxy%20also%20ships%20with%20a,%2C%20backend%2C%20and%20server%20basis.[Último acceso: 2020]

[4] Clustering. LXC system container manager [en línea]. Disponible en https://lxd.readthedocs.io/en/latest/clustering/

[5] ¿Cómo funciona “cat << EOF" en bash? [En línea]. Disponible en https://www.dokry.com/5862

[6] 31 + ejemplos para el comando sed [En línea]. Disponible en https://likegeeks.com/es/sed-de-linux/

[7] Michael Iatrou. LXD Clusters: A Primer [En línea]. Disponible en https://ubuntu.com/blog/lxd-clusters-a-primer

[8] Setting up an LXD Cluster [En línea]. Disponible en https://osm.etsi.org/docs/user-guide/16-lxd-cluster.html