Microproyecto 1: Cluster LXD, Balanceador de Carga HAProxy

Edison Orozco
Esp. Inteligencia Artificial
U. Autónoma de Occidente

Juan Pablo Grisales
Esp. Big Data
U. Autónoma de Occidente

***Abstract*—The objective of this micro-project is to practice the concepts of containers, clusters, load balance and provisioning using tools like LXD, HAProxy, Apache to understand how works a request made by a user to application in the cloud.**

Keywords—cluster, load balancer, container, virtual machine, web server, haproxy

I. INTRODUCCIÓN

Con este microproyecto de la clase de Computación en la Nube se pretende poner en práctica los conceptos de clustering, balanceo de carga.

II. CLUSTER

A. Creación de las máquinas virtuales

Como parte fundamental para este desarrollo, se debían crear tres máquinas virtuales con nombres asignados ubuntuvm1, ubuntuvm2, ubuntuvm3. En cada una de ellas con un contenedor, web1, web2 y haproxy. Para los dos primeros instalamos el servidor Apache en cada uno y para el ultimo el balanceador de carga HAProxy ejecutando el algoritmo de balanceo Round Robin, que básicamente consiste en asignación de peticiones pro turnos a los servidores.

Las maquinas virtuales se crearon a partir de un Vagrantfile como el que se muestra:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

ENV['VAGRANT_NO_PARALLEL'] = 'yes'

Vagrant.configure(2)do |config|

  NodeCount = 3

   (1..NodeCount).each do |i|
     config.vm.define "ubuntuvm#{i}" do |node|
        node.vm.box = "ubuntu/bionic64"
        node.vm.hostname = "ubuntuvm#{i}.example.com"
        node.vm.network "private_network", ip: "192.168.100.10#{i}"
        node.vm.provider "virtualbox" do |v|
           v.name = "ubuntuvm#{i}"
           v.memory = 1024
           v.cpus = 1
         end
      end
    end

end
```

B. Clúster

Para la creación del clúster iniciamos instalando LXD a través del terminal con el comando ***sudo snap install lxd***. Posterior a esto iniciamos con la creación del clúster propiamente con el comando ***lxd init*** y contestamos las preguntas indicando que vamos a hacer clustering. Para la primera maquina debemos indicar que es un nuevo clúster. Para el resto debemos indicar que vamos a agregar a un clúster existente indicando la IP de un elemento existente. Cuando hayamos terminado tendremos lo siguiente:

```
vagrant@ubuntuvm1:~$ lxc cluster list
+-----------+----------------------------+----------+--------+-------------------+
|   NAME    |            URL             | DATABASE | STATE  |      MESSAGE      |
+-----------+----------------------------+----------+--------+-------------------+
| ubuntuvm1 | https://192.168.100.101:8443 | YES      | ONLINE | fully operational |
+-----------+----------------------------+----------+--------+-------------------+
| ubuntuvm2 | https://192.168.100.102:8443 | YES      | ONLINE | fully operational |
+-----------+----------------------------+----------+--------+-------------------+
| ubuntuvm3 | https://192.168.100.103:8443 | YES      | ONLINE | fully operational |
+-----------+----------------------------+----------+--------+-------------------+
```

C. Contenedores

Una vez creado el contenedor, se deben crear los contendedores en las maquinas virtuales. En dos de ellas se instalaron servidores web (Apache) y en la tercera se instaló el balanceador de carga HAProxy, de la siguiente manera: ubuntuvm1=uweb1, ubuntuvm2=uweb2 y ubuntuvm3=haproxy. Estos contenedores se instalaron ejecutando el comando ***lxc launch ubuntu:18.04 uweb1*** para el caso del contenedor localizado en la maquina ubuntuvm1. Para los otros se ejecutó la misma acción cambiando el nombre del contenedor. Con el comando ***lxc list*** podemos comprobar que los contenedores están creados y corriendo.

```
vagrant@ubuntuvm1:~$ lxc list
+---------+---------+----------------------+------+------------+-----------+----------+
|  NAME   |  STATE  |         IPV4         | IPV6 |    TYPE    | SNAPSHOTS | LOCATION |
+---------+---------+----------------------+------+------------+-----------+----------+
| haproxy | RUNNING | 240.103.0.47 (eth0)  |      | PERSISTENT | 0         | ubuntuvm3 |
+---------+---------+----------------------+------+------------+-----------+----------+
| uweb1   | RUNNING | 240.101.0.96 (eth0)  |      | PERSISTENT | 0         | ubuntuvm1 |
+---------+---------+----------------------+------+------------+-----------+----------+
| uweb2   | RUNNING | 240.102.0.218 (eth0) |      | PERSISTENT | 0         | ubuntuvm2 |
+---------+---------+----------------------+------+------------+-----------+----------+
```

D. Servidores web y balanceador de carga

Luego de tener listos los contenedores instalamos los servidores web y el balanceador de carga. Para los primeros ingresamos a cada contenedor mediante el comando ***lxc exec uweb1 /bin/bash*** y estando ahí ejecutamos ***apt install apache2.*** Para el balanceador de carga ingresamos al contendor de la misma manera antes mencionada y para este caso ejecutamos ***apt install haproxy*** para instalar el balanceador de carga, que como mencionamos anteriormente ejecuta el algoritmo de balanceo Round Robin.

Podemos comprobar las instalaciones mencionadas ejecutando el comando de estado ***systemctl status***. A continuación, se muestra la comprobación:

```
vagrant@ubuntuvm1:~$ lxc exec uweb1 /bin/bash
root@uweb1:~# systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
    Drop-In: /lib/systemd/system/apache2.service.d
             └─apache2-systemd.conf
     Active: active (running) since Wed 2021-11-03 03:13:58 UTC; 10min ago
    Process: 197 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 259 (apache2)
      Tasks: 55 (limit: 1150)
     CGroup: /system.slice/apache2.service
             ├─259 /usr/sbin/apache2 -k start
             ├─260 /usr/sbin/apache2 -k start
             └─261 /usr/sbin/apache2 -k start

Nov 03 03:13:56 uweb1 systemd[1]: Starting The Apache HTTP Server...
Nov 03 03:13:58 uweb1 systemd[1]: Started The Apache HTTP Server.
```

```
vagrant@ubuntuvm1:~$ lxc exec uweb2 /bin/bash
root@uweb2:~# systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
    Drop-In: /lib/systemd/system/apache2.service.d
             └─apache2-systemd.conf
     Active: active (running) since Wed 2021-11-03 03:14:08 UTC; 11min ago
    Process: 195 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 262 (apache2)
      Tasks: 55 (limit: 1150)
     CGroup: /system.slice/apache2.service
             ├─262 /usr/sbin/apache2 -k start
             ├─264 /usr/sbin/apache2 -k start
             └─265 /usr/sbin/apache2 -k start

Nov 03 03:14:05 uweb2 systemd[1]: Starting The Apache HTTP Server...
Nov 03 03:14:08 uweb2 systemd[1]: Started The Apache HTTP Server.
```

```
vagrant@ubuntuvm1:~$ lxc exec haproxy /bin/bash
root@haproxy:~# systemctl status haproxy
● haproxy.service - HAProxy Load Balancer
   Loaded: loaded (/lib/systemd/system/haproxy.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2021-11-03 03:14:18 UTC; 11min ago
     Docs: man:haproxy(1)
           file:/usr/share/doc/haproxy/configuration.txt.gz
  Process: 228 ExecStartPre=/usr/sbin/haproxy -f $CONFIG -c -q $EXTRAOPTS (code=exited, status=0/SUCCESS)
 Main PID: 238 (haproxy)
    Tasks: 2 (limit: 1150)
   CGroup: /system.slice/haproxy.service
           ├─238 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid
           └─243 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid

Nov 03 03:14:17 haproxy systemd[1]: Starting HAProxy Load Balancer...
Nov 03 03:14:18 haproxy haproxy[238]: Proxy web-backend started.
Nov 03 03:14:18 haproxy haproxy[238]: Proxy web-backend started.
Nov 03 03:14:18 haproxy haproxy[238]: Proxy http started.
Nov 03 03:14:18 haproxy haproxy[238]: Proxy http started.
Nov 03 03:14:18 haproxy systemd[1]: Started HAProxy Load Balancer.
```

E. Aprovisionamiento

Para el desarrollo del aprovisionamiento, haciendo uso de Vagrant, es necesario generar los comandos iniciales para instalar e iniciar las maquinas en las cuales se va a trabajar como se ve a continuación:

```
Vagrant.configure("2") do |config|

  if Vagrant.has_plugin? "vagrant-vbguest"
    config.vbguest.no_install = true
    config.vbguest.auto_update = false
    config.vbguest.no_remote = true
 end

    #creo la maquina servidor
    config.vm.define :ubuntuvm1 do |ubuntuvm1|
        ubuntuvm1.vm.box = "bento/ubuntu-20.04"
        ubuntuvm1.vm.hostname = "ubuntuvm1"
        ubuntuvm1.vm.network :private_network, ip: "192.168.100.101"
        ubuntuvm1.vm.provision "shell", path: "script1.sh"
     end

    #creo la maquina vm2
    config.vm.define :ubuntuvm2 do |ubuntuvm2|
        ubuntuvm2.vm.box = "bento/ubuntu-20.04"
        ubuntuvm2.vm.hostname = "ubuntuvm2"
        ubuntuvm2.vm.network :private_network, ip: "192.168.100.102"
        ubuntuvm2.vm.provision :shell, path: "script2.sh"
     end

    #creo la maquina vm3
    config.vm.define :ubuntuvm3 do |ubuntuvm3|
        ubuntuvm3.vm.box = "bento/ubuntu-20.04"
        ubuntuvm3.vm.hostname = "ubuntuvm3"
        ubuntuvm3.vm.network :private_network, ip: "192.168.100.103"
        ubuntuvm3.vm.provision :shell, path: "script3.sh"
     end
end
```

dentro de ese código es necesario dar el comando de aprovisionamiento, es en el cual se asignaran los comando para instalar los aplicativos requeridos para el funcionamiento de nuestro cluster, para ello es necesario como primera medida inicializar los LXD y cargar en la primera máquina llamada ubuntuvm1 la cual en nuestro caso hará las veces de servidor, entonces es este el encargado de contener el clúster y contener el HAProxy, quien es el encargado de realizar el balanceo de las peticiones solicitadas por el host:

```
cat <<EOF | lxd init --preseed
config:
  core.https_address: 192.168.100.101:8443
  core.trust_password: admin
networks:
- config:
    bridge.mode: fan
    fan.underlay_subnet: 192.168.100.0/24
  description: ""
  name: lxdfan0
  type: ""
storage_pools:
- config: {}
  description: ""
  name: local
  driver: dir
profiles:
- config: {}
  description: ""
  devices:
    root:
        path: /
        pool: local
        type: disk
    eth0:
      name: eth0
      network: lxdfan0
      type: nic
cluster:
    server_name: ubuntuvm1
    enabled: true
    member_config: []
    cluster_address: ""
    cluster_certificate: ""
    server_address: ""
    cluster_password: ""
```

Se debe crear un certificado, que es una llave única, con las cual las otras máquinas se van a poder conectar al servidor como se muestra:

```
echo "genero el certificado"
sed ':a;N;$!ba;s/\n/\n\n/g' /var/snap/lxd/common/lxd/cluster.ctr > /vagrant/cluster.txt
```

Dentro de esta máquina se debe modificar el archivo de haproxy.cfg, el cual es el encargado de contener un Front End (la interfaz que interactúa con el usuario final) y un Back End (el cual le indica a la maquina como alternar entre los servidores web en este caso apache montados en las maquinas Ubuntuvm2 y 3.

```
backend web-backend
  balance roundrobin
  stats enable
  stats auth admin:admin
  stats uri /haproxy?stats
  server web1 ubuntuvm2:80 check
  server web2 ubuntuvm3:80 check

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

Así mismo dentro de esta máquina se configuran los mensajes de error 503 cuando el servidor se encuentre fuera de servicio.

Para las demás maquinas, como el clúster ya ha sido creado, solamente se debe validar el certificado y cargar los servidores Apache, adicionalmente se debe modificar el contenido WEB que se desea mostrar

```
cat <<EOF | sudo lxd init --preseed
config: {}
networks: []
storage_pools: []
profiles:[]
cluster:
    server_name: ubuntuvm2
    enable: true
    member_config: []
      - entity: storage_pool
        name: local
        key: source
        value: ""
        description: '"source" property for storage pool "local"'
    cluster_address: 192.168.100.101:5180
    cluster_cartificate: "$certificado"
    server_adress: 192.168.100.102:8443
    cluster_password: admin
    cluster_certificate_path: ""
EOF
```

BIENVENIDO SERVIDOR WEB1

BIENVENIDO SERVIDOR WEB2

III. REFERENCES

[1] Linuxcontainer.org. Clustering. https://linuxcontainers.org/lxd/docs/master/clustering

[2] Youtube.com. LXD Clustering Explained - Part 1. https://www.youtube.com/watch?v=LhF2HxqKPzk&t=344s

[3] Youtube.com. LXD Clustering Explained - Part 2. https://www.youtube.com/watch?v=LhF2HxqKPzk&t=344s.

[4] Linuxcontainer.org. Containers. https://linuxcontainers.org/lxd/docs/master/containers

[5] www.ochobitshacenunbyte.com. Balanceo de carga con HAProxy. https://www.ochobitshacenunbyte.com/2019/12/17/balanceo-de-carga-con-haproxy-en-ubuntu-18-04/

[6] O. Mondragón, Práctica Linux Containers.

[7] O. Mondragón, Práctica Aprovisionamiento.

[8] O. Mondragón, Balanceo de carga con HAProxy.