Installations Keepalived mit HAProxy Loadbalancer mit Floating IPs mittels Ansible script

Beschreibung:

Auf der vorherigen Seite haben wir gesehen wie wir den manuell installieren und wie das ganze funktioniert.
Und wie Services eingebunden werden.

Hier ein Ansible Beispiel wie ein Loadbalancer mit floating ips bei Hetzner aufgesetzt werden aknn.
Ziel ist ein Loadbalancer zu haben der selbst eine Floating ips zwischen beiden LBs hat und keine andere Aufgabe hat als wenn ein LB offline geht die floating ips auf den oder die weiteren LB auszurollen.
Wofür könnte man sowas brauchen. Zum Beispiel umd diese floating ips zu Tiunneln odewr immer einen VPN Server der auf den LBs laufen könnte erreichbar zu halten.
Für was auch immer. Der Loadbalancer in diesem Setup ist Dualstack fähig also ipv4 und ipv6

Hier geht es ja nur darum, einen LB zu haben der Floating ips von links nach recht schiebt und umgegehrt.
Aber wenn man noch Services anglegen will, das die dann umgeschwenkt werden, kann man in der services.yml auch noch festlegen.

Ein internes Netz zwischen den Loadbalancern für die Heartbeat funktion. (Hier wird ein broadcast gesendet, der über Öffentliche Netzte nicht funktioniert)

Verwendenung des ansible Scriptes

Aufbau des Verzeichnisses:

```
.
├── getids.sh
├── hetzner.yml
├── hosts.ini
├── installkeepalived.yml
├── installpython3.yml
├── installwireuardmesh.yml
├── install_yq_jq.sh
├── output.txt
├── README.md
├── server_table.txt
├── services.yml
├── table.txt
├── templates
│   ├── haproxy.cfg.j2
│   ├── keepalived.j2
│   ├── keepalived_state_change.j2
│   └── wireguard.conf.j2
├── testapache2erstellen
│ ├── hosts.ini
│   └── installapache2withindex.yml
└── update_hetzner.sh
```

getids.sh

ist ein Script mit der die hetzner.yml erweitert wird.
Einfach ausführen auf dem Service System.
Mit den jeweiligen Parametern wird die hetzner.yml erweitert.
Bei ipv4 wird immer die einzelne Adresse vergeben, bei ipv6 wird immer ein komplettes Subnetz vergeben.

Die Parameter:

```
Usage: ./getids.sh [OPTIONS]
Options:
  -h, --help               Show this help message
  -d, --debug              Show debug information (JSON output)
  -lfl LABEL, --loadbalancer-floating-label LABEL
                           Specify label for loadbalancer floating IPs
```

```
  -fl LABEL, --floating-label LABEL
                        Specify label for other floating IPs
```

mit -h wird die hilfe aufgerufen

mit -lfl oder --loadbalancer-floating-label wird festgelegt das diese ips für den Loadbalancer selbst verwendet werden sollen.
Sprich diese werden dann auch einer Netzwerkkarte hinzugefügt Welche das ist wird in der Hostvariable

```
interface_floating=eth0
```

festgelegt
Diese werden dann als erstes in die hetzner xml eingetragen. sollten zwei ipvs mit dem selben Label vorhanden sein. Entscheidet der Zufall darüber, welche genommen wird. Deshalb immer auf eindeutige Label bzw Label Kombinationen achten.
Label können via Komma separiert werden um mehre Labels anzugeben. Es müssen alle Labels erfüllt sein, damit sie genommen werden (und verknüpft)

Beispiel:

```
-lfl lb_ips,loadbalancer_oldenburg
```

 -fl LABEL, --floating-label LABEL, Definiert die Floating IPS die auf dem an den Loadblancer gepackt werden sollen, aber keiner Netzwerkkarte zugeordnet, da diese eventuell weitergetunnelt werden könnten, oder anderweitig verwendet werden sollen.
SIe sollen nur auf dem Loadbalancer zur Verfügung stehen.

Es wird vom sSript auch geprüft ob die IP schon als IP für den Loadbalancer vergeben wurde, um doppel Vergebung zu vermeiden.

Beispiel:

```
-fl loadbalancer_oldenburg
```

hetzner.yml

Diese Datei enthält einmal den API-KEY und die URL für für Hetzner.
Sollte diese Datei nicht existieren. Einfach anlegen und diese Sektionen einfügen.
Damit ist auch das Grundgerüst gleich erklärt
Einmal ein Global Bereich für die Variablen und der floating IP-Bereich in dem die Floating IPs gespeichert werden.

```
global:
  hetzner_api_key: zHE7y9vFJQfWEpSuSUhRynLRScO5yM2OLA85NRAswnf23bFDZkugj7LORwVQq5fp
  hetzner_api_url: https://api.hetzner.cloud/v1
floating_ips:
```

Wenn das getids.sh script ausgeführt wurde, dann sieht die hetzner.yml als Beispiel so aus:

```
floating_ips:
- name: "lbs_ipv6"
  hetzner_id: xxxxx
  ipv6: "2xxxxxxxxxx::/64"

- name: "lbs_ipv4"
  hetzner_id: 61xxxx
  ipv4: "78.46.xxx.xxx"

- name: "ipv4_free04"
  hetzner_id: 16xxxx
  ipv4: "116.202.1xxx.xxx"

- name: "ipv4_free03"
  hetzner_id: 2xxxx
  ipv4: "116.202.xxx.xxx"

- name: "ipv6_free04"
  hetzner_id: 4xxxxx
  ipv6: "xxxxxxxxxxx::/64"
```

Das Ansible script wertet die ersten beiden floating IPs als Eigene IPs für sich selbst.
Das getids.sh script baue die hetzner.yml so das der erste wert eine ipv4 und der zweiter eine ipv6 aus der liste ist die dem Label entsprechen.

In der Hetzner.yml wird der Name der ip bei hetzner gespeichert
die id und der typ steht vor der ip entweder ipv4 oder ipv6.

alle anderen floating ips werden dem Loadbalancer nicht zugewiesen (also an ein interface gebunden) sondern werden nur auf der Machine verfügbar gemacht, so das Sie an ein Interface gebunden werden könnten.
Nun können diese Floating IPs flexibel verwendet werden. Entweder werden sie an Interfaces gebunden oder weiter getunnelt.
Je nach dem was man möchte.

host.ini

Sie enthält die Loadbalancer auf die das Script angewendet werden soll

In Ihr ist eine Gruppe für die dann gloabale variablen gelten.
Denn die müssen bei allen Loadblancern gleich sein und bei jedem host muss noch die hetzner_serverid angegben werden. Damit dann beim failover die ips an den richtigen server gepackt werden.
desweiteren muss noch das interface für die flaoting ip angeben werden auf den der Loablancer listen soll.
und ein weiteres interface mit einem privaten netzwerk über das das heratbeat laufen soll.
In dieser Beispiel host.ini sind es 3 Loadbalancer instancen. Minimum müssen zwei angeben sein.

Es gibt unter global variables der Pfad zur keepalived_state_change.sh
Des weiteren gibt es eine virtual_router id, die muss auch bei allen gleich sein, denn die beschreibt die Gruppe, diese ID wird per Broadcast gesendet.
Der Paramater **balancer_passphrase=** ist veraltet und wird nicht mehr unterstützt, da konnte noch ein Password vergeben werden. Deshalb sollten diese Verbindungen entweder durch einen Tunnel (Dafür der Wiregaurd, wenn man möchte) oder eine dedizierte private Verbindung in einem VLAN oder wie bei Hetzner ein lokales Netz.
Der Name ist frei vergebbar für die Datei und auch der Name, denn die Datei wird vom Ansible script erstellt.

```
[loadbalancer]
host1 hetzner_id=1020301 interface_floating=eth0 interface_heartbeat=eth1 ansible_ssh_user=root
host2 hetzner_id=1020302 interface_floating=eth0 interface_heartbeat=eth1 ansible_ssh_user=root
host3 hetzner_id=1020303 interface_floating=eth0 interface_heartbeat=eth1 ansible_ssh_user=root


[loadbalancer:vars]
virtual_router_id=50
script_path=/root/keepalived_state_change.sh
```

installkeepalived.yml

Dieses script rollt die Loadbalancer aus.
es benötigt keine weiteren paramter als die host.ini

```
ansible-playbook -i host.ini installkeepalived.yml
```

Und schon rennt er los.

installpython3.yml

Dieses script installiert die python3 library uf den hosts
Es ist ein ansible script.

```
ansible-playbook -i host.ini installpython3.yml
```

Muss nur einmal gemacht werden

installwireuardmesh.yml

Wenn der/die Loadbalancer als Wiregaurd Meshserver dienen soll, dieses Playbook ausführen.
wenn in den Hostvariablen diese Variable angeben ist und true ist, wird dieses palybook darauf angewandt
Es wir ein reines internes VPN-Netz gebaut. Und somit nur routen über das lokale subnetz gebaut.
Siehe variable wireguard_subnet.
Diese kann dazu verwendet werden um das keepalived heartbeat über einen Tunnel laufen zu lassen, wenn kein lokales Netzwerk zwischen den Loadbalancern möglich ist. Dann wird in der hostvariable für das Loadbalancer Netz der name des Wirguardadapters angegeben. In unserem Fall wirelb siehe weiter unten gloabale Wireguard Variablen.

```
enable_wireguard=true
```

Desweiteren werden folgende globale Variablen angewandt.
Die host.ini damit dann erweitern in der section [loadbalancer:vars]

```
wireguard_subnet=10.15.2.0/24
wireguard_interface_name=wirelb
wireguard_port=51820
wireguard_conf_dir=/etc/wireguard
```

install_yq_jq.sh

Dieses script installiert die Abhängihketen von yq und jq um json files zu parsen.
Diese json files kommen von der hetzner API.
Einfach ausführen mit root rechten auf dem Service System.
Dies sind abhängigkeiten die für getids.sh benötigt werden.

Muss nur einmal gemacht werden

output.txt

Enthält wenn die getids.sh mit dem parameter debug aufgerufen wird die komplette hetzner.json stream.
Dient zum debuggen.

server_table.txt

Diese enthält die Liste der Server mit den ids.
Diese werden erstellt wenn getids.sh ohne Paramter aufgerufen wird.
Gleichzeitg wird diese Tabelle auch in der Console nochmals ausgegben

services.yml

Über diese Datei werden, die Dienste (server festgelet die Überwacht werden sollen. Webserver smtp server etc.
Diese wird benötigt wenn er nicht nur als Loadbalancer für die IPS sondern auch für services dienen soll.

Aufbau:

```
services:
  - name: "nextcloud"
    floating_ipv4: "198.51.xxx.xxx"
    floating_ipv6: "xxxxxxxx"
    frontend_port: 443
    frontend_protocol: "TCP"
    backend_servers:
      - name: "nextcloud01"
        ipv4: "192.0.xxx.xxx"
        ipv6: "2001:xxxxxx"
        backend_port: 443
        backend_protocol: "TCP"
      - name: "nextcloud02"
        ipv4: "192.0.xxx.xxx"
        ipv6: "2001:xxxx.xxxx"
        backend_port: 443
        backend_protocol: "TCP"
```

Es können beliebig viele Services mit backend server angelegt werden.
Eigentlich selbst erkärend.

-name des dienstes
darunter die beiden floating ips
darunter der port und protokoll auf LB Seite ankommend

dann die backend server als liste
wieder name ipv4 und ipv6 des server port und Protokoll
und dannd er nächste server.
Dann könnet mann den den nächsten Dienst uim gleichen format anlegen.

Möchte man gar keine weiteren Dienste lässt man in der yml Datei nur services: stehen

```
services:
```

table.txt

Diese enthält die Liste der floating ips und den Servern mit den ids.
Diese werden erstellt wenn getids.sh ohne Paramter aufgerufen wird.
Gleichzeitg wird diese Tabelle auch in der Console nochmals ausgegben

templates

Im Template Verzeichnis sind die Template Dateien

```
├── templates
│   ├── haproxy.cfg.j2
│   ├── keepalived.j2
│   ├── keepalived_state_change.j2
│   └── wireguard.conf.j2
```

haproxy.cfg.j2 ist das template das die haproxy config aus der service.xml baut
Die keepalived.j2 dient zur erstellung der keepalived config
Die keepalived_state_change.j2 dient zur erstellungen des scriptes, in diesem Falle für Hetzner zum ip wchsel beim Loadbalancer wechsel bzw. failover. DIese script Dateien werden für das ansible Playbook benötigt.
wireguard.conf.j2 erstellt die wireguard konfiguration. Diese wird vom wireguardplaybook verwendet

Test Apache2

Der Testapache dient dazu schnell Apache server auszurollen die eine http webseite bereitstellen mit Hostname und ip Adresse in der index.html. Dient zum testen des keepalived mit dem haproxy. Dann kann in der service.xml diese hosts angeben werden.

```
└── testapache2erstellen
    ├── hosts.ini
    └── installapache2withindex.yml
```

hosts.ini für die server zum installieren
installapache2withindex.yml erstellt die index.html

update_hetzner.sh

Diese datei ist nur ein hilfe um nicht immer wieder den gleichen Befehl eintippen zu müssen wenn die Tags feststehen

Inhalt

```
#!/bin/bash
./getids.sh -lfl lb_ips,loadbalancer_oldenburg -fl loadbalancer_oldenburg
```

Ausführbar machen

```
chmod +x update_hetzner.sh
```

Abschluss

Damit wäre die installation unseres Ansible Script Ordner eingerichtet, alle abhängigkeiten uaf unserem Service PC installiert und die Dateien erklärt.
Wichtig ist das auf den LB Servern vorher noch das installpython3.yml ansible playbook ausgeführt wird um auch Sicher zu gehen das python3 zur verfügung steht auf den Servern.

Version #16
Erstellt: 2 Dezember 2023 10:35:24 von Admin
Zuletzt aktualisiert: 30 Mai 2024 13:06:28 von Admin