

# Installation und einrichtung

## Beschreibung:

Hier ist ein Tutorial, um zwei Loadbalancer mit einer Floating-IP erreichbar zu machen. Damit nicht nur die Backendserver-Verbindung hochverfügbar ist, sondern auch die Verbindungs-IP (Floating-IP) mithilfe von zwei Loadbalancern anstelle von einem. Zusätzlich überprüfen die Loadbalancer die OpenVPN-Dienste und HTTPS-Dienste.

Es kann natürlich jeder anderer Dienst in betracht gezogen werden

## Hinweis:

Bei einem N2N Tunnel funktioniert der LVS nicht richtig, da dann haproxy nehmen. Aber als reiner LB für die floating IPS reicht es.

## Vorraussetzungen:

5 x Public IP Adresses, 2 x LB , 2 x RVS and 1 x Floating ip

2 x Machines as LB01 and LB02, with Linux (eg Debian) and ssh access

2 x Machines as RVS01 and RVS02 for example

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

In unserer Testumgebung haben wir diese IPS

Floating IP : 49.12.154.74

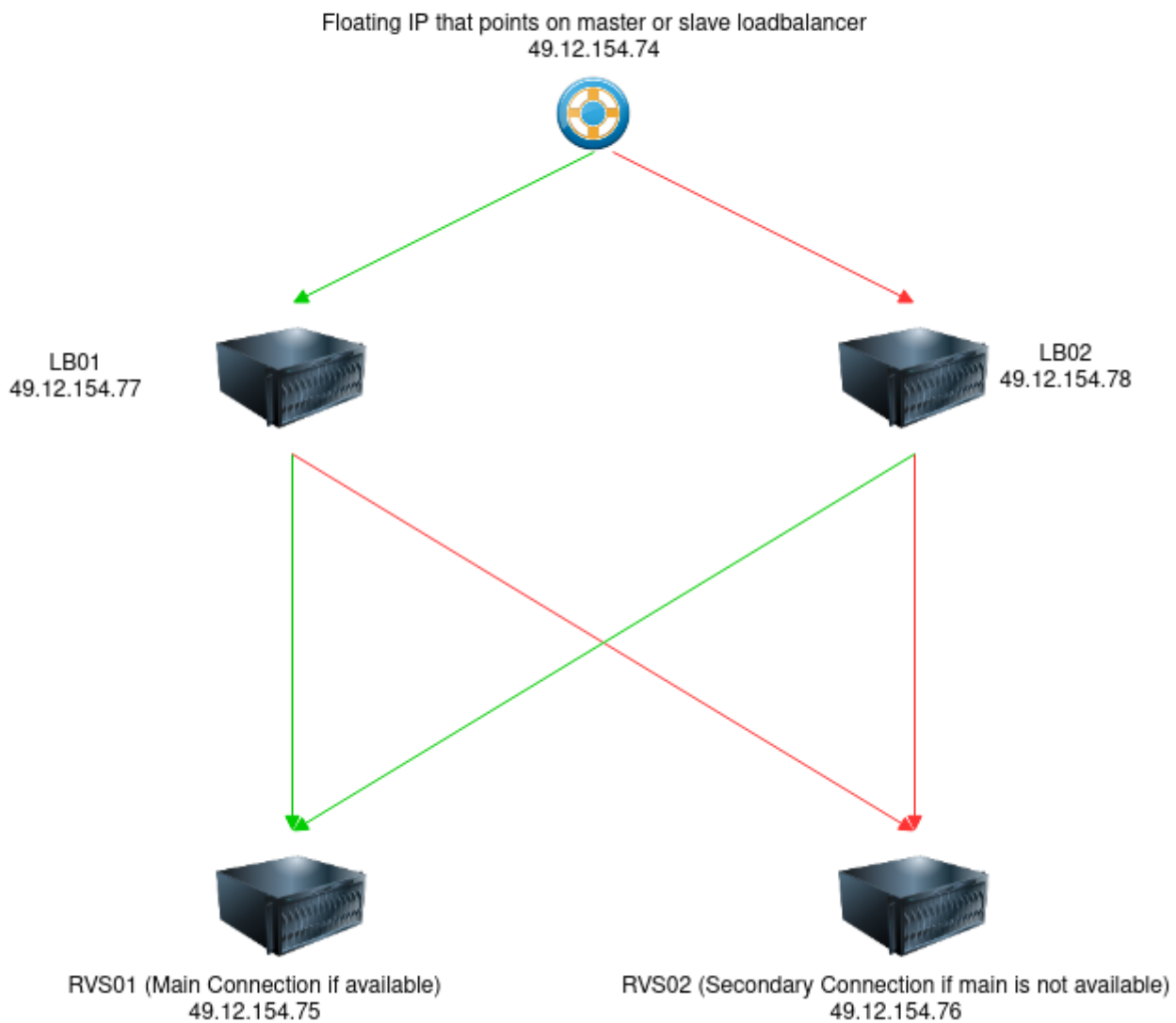
LB01 : 49.12.154.77

LB02 : 49.12.154.78

RVS01 : 49.12.154.75

RVS02 : 49.12.154.76

- Master Connection (Regular connection)
- Slave Connection (Failover connection)



## Installation:

Installiere die Software-Anforderungen durch die folgenden Pakete und Konfigurationen auf LB01 und LB02. Es ist notwendig, dass die Netzwerk-IP und SSH korrekt konfiguriert sind für die nächsten Schritte...

```
apt update
apt dist-upgrade
apt install curl wget
apt install keepalived
apt install python3 python-is-python3
```

Das Routing muss in der sysctl auch aktiviert sein.

```
nano /etc/sysctl.conf
```

Inhalt am Ende anfügen

```
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
```

nun anwenden

```
sysctl -p
```

Anforderungen an die RVS-Server für HA mit Loadbalancer und Floating-IP.

Wir müssen nur eine iptables-Regel hinzufügen, weil die RVS-Server zur Floating-IP umgeleitet werden müssen.

Befehl zur direkten Ausführung. Siehe Punkt 6, um dies automatisch beim Start des Servers durchzuführen.

```
iptables -t nat -A PREROUTING -d 49.12.154.74 -j REDIRECT
```

Keepalived-Konfiguration, der lustige Teil.

Wir beginnen mit der Erklärung der Konfiguration von Server 1, die Konfiguration von Server 2 ist fast identisch mit der von Server 1, mit nur wenigen Änderungen.

## Wir starten mit Server 1.

**Hinweis:** Hinweis: In version 3 von Keepalived wurde die Authentifizierung.

Der Abschnitt .

```
...
authentication {          #Validierungsinformationen einstellen, zwei Knoten müssen konsistent
    sein
    auth_type PASS        #Authentifizierungstypen einrichten, hauptsächlich PASS und AH
    auth_pass 8nzlTBSoSrpJP0i77TgL!    #Das Authentifizierungspasswort einstellen, zwei
    Knoten müssen unter einem vrrp_instance dasselbe Passwort verwenden

}
```

....

kann entfernt werden.

Denn dieser wird ignoriert.

**Das bedeutet das für die Heartbeat Verbindung kein Öffentliches Netz mehr gewählt werden sollte, sondern nur Private / dedizierte Verbindungen.**

**Hinweis:** Keepalived Heartbeat ist Standard Multicast, sollte die Verbindung zwischen den LBs kein Multicast unterstützen

dann den Abschnitt weiter unten Unicast anschauen. Ich würde wahrscheinlich generell Unicast verwenden um Fehler vorzubeugen.

Erstellen Sie eine neue Datei `/etc/keepalived/keepalived.conf`.

Die Datei hat verschiedene Abschnitte:

`global_defs{}`: Dies enthält Benachrichtigungen und Identifikationen der Loadbalancer.

`vrp_instance VI_1{}`: Enthält die Verbindungsauthentifizierung und die Zuweisung der virtuellen IP (Floating IP). Hier können wir auch Skripte definieren, die bei Statusänderungen von Master zu Backup oder von Backup zu Master ausgeführt werden.

Zum Beispiel, um eine andere Floating-IP von Hetzner über ein sh-Skript zuzuweisen.

`virtual_server{}`: Hier wird die virtuelle Server-IP mit Port und Protokoll (UDP, TCP) definiert.

In diesem Abschnitt ist der letzte Abschnittstyp eingebettet.

Die `real_servers`, die in Punkt d aufgelistet sind.

`real_server{}`: Hier werden die RVS-Server mit Service-Checks definiert.

## Die Global Config

```
#=====Global Config=====#

#Email notification
global_defs {
notification_email {
root@localhost //Accept email address
}

#Mailserver Settings
notification_email_from keepalived@localhost //Mailing address
smtp_server 127.0.0.1 //Send mail server IP
smtp_connect_timeout 30 //Mail connection timeout
#Rounting settings #here master on the seconday lb02 it must called be slave
router_id master #An identity that identifies the keepalived server running (type an string)
}

#===== End Globals =====#
```

`vrp_instance` section with virtual ip

Die Authentication wurde in Version 3 entfernt. Da das Hauptmerkmal darauf gelegt wurde, dass die interne Verbindung für das Heartbeat sicher ist. Sprich Firewall regeln dediziertes VPN etc.

```
#===== HA for Floating IP =====#
vrrp_instance VI_1 {          #VRRP instance definition section
    state MASTER              #Gibt an, dass der Knoten der Hauptknoten ist (Großbuchstaben) und der Standby-
Knoten BACKUP ist
    interface ens18          #Netzwerkschnittstelle, über die die Interne Kommunikation laufen soll, heartbeat.
Muss ein Privates Netzwerk sein, da hier ein Broadcast gesendet wird.
    virtual_router_id 51     #VRRP Gruppenname, zwei Knoten müssen gleich eingestellt werden, um anzuzeigen,
dass jeder Knoten derselben VRRP-Gruppe angehört. Muss eine ganze Zahl sein
    priority 100             #Priorität des Hauptknotens (1-254), Standard 100, beachten Sie, dass die Priorität des
sekundären Knotens niedriger sein muss als die des Hauptknotens
    advert_int 1             #Legt das Zeitintervall zwischen Synchronisationsprüfungen zwischen zwei Knoten fest,
die beiden Knoten müssen konsistent sein. Wert in Sekunden
    authentication {        #Validierungsinformationen einstellen, zwei Knoten müssen konsistent sein
        auth_type PASS      #Authentifizierungstypen einrichten, hauptsächlich PASS und AH
        auth_pass 8nzITBSorJP0i77TgL! #Das Authentifizierungspasswort einstellen, zwei Knoten müssen
unter einem vrrp_instance dasselbe Passwort verwenden
    }

    virtual_ipaddress {     #Virtuelle IP (Floating IPs) angeben, beide Knoten müssen gleich eingestellt sein, es
kann mehr als eine geben, eine pro Zeile. Als Parameter dev das Netzwerkgerät, dem die IP zusätzlich
zugewiesen wurde
        49.12.154.74 dev ens18
    }

#Benachrichtigungsbereich, wann welches Skript im angegebenen Zustand ausgeführt wird (z.B. Skript zur
Zuweisung von Floating IP per Skript bei Hetzner oder anderen)
#Nur notify wird bei jedem Zustand aufgerufen.
#Aber die folgenden Parameter werden übergeben
#notify /pfad_zum_skript/skript_allezustände.sh
#Im Skript allezustände gibt es 3 Parameter, die überprüft werden können
#Type enthält den Wert "GROUP" oder "INSTANCE"
#TYPE = $1
#Name enthält den Namen der Gruppe oder Instanz
#NAME = $2
#State enthält den Zustand. Diese sind "MASTER" "BACKUP" "FAULT" mit * kann man einen unbekanntem
Zustand abfangen
#STATE = $3
```

#Beispiel skript\_allezustände.sh weiter unten im nächsten Codekasten.

Da wir ein script für alle zustände haben, wenn wir das hetzner script verwenden möchten dann müssen wir alle scripte unten austauschen,

```
#ansonsten selbst andere scripte definieren
```

```
#Diese rufen die Skripte im gegebenen Zustand auf, daher ist keine Überprüfung im sh-Skript erforderlich
```

```
#notify_master /pfad_zum_skript/skript_hauptknoten.sh
```

```
# (oder notify_master " /pfad_zum_skript/skript_hauptknoten.sh)
```

```
#notify_backup /pfad_zum_skript/skript_backup.sh
```

```
# (oder notify_backup " /pfad_zum_skript/skript_backup.sh)
```

```
#notify_fault /pfad_zum_skript/skript_fehler.sh
```

```
# (oder notify_fault " /pfad_zum_skript/skript_fehler.sh)
```

```
}
```

```
#=== End Ha for Floating IP =====#
```

Beispiel skript\_allezustände.sh mit Hetzner Floating-IP-Zuweisungsbeispiel. In unserer Textumgebung verwenden wir keine Hetzner Floating-IP, aber als Beispiel für andere, die es nutzen werden. Dies ist ein Beispiel-Skript für lb01, daher wird die Hetzner Server-ID 5605626 verwendet. das script wertet über Parameter \$3 also state aus, was Phase ist, master, backup ,unknown

```
#!/bin/bash
```

```
#lb01 ID : 5605626
```

```
#lb02 ID : 5605974
```

```
#floatingipv4 id =: 245713
```

```
#floatingipv6 id =: 246612
```

```
TYPE=$1
```

```
NAME=$2
```

```
STATE=$3
```

```
case $STATE in
```

```
    "MASTER") curl -X POST -H "Content-Type: application/json" -H "Authorization: Bearer <hetzner-api-key>" -d '{"server": 5605626}' 'https://api.hetzner.cloud/v1/floating_ips/245713/actions/assign'
```

```
        curl -X POST -H "Content-Type: application/json" -H "Authorization: Bearer <hetzner-api-key>" -d '{"server": 5605626}' 'https://api.hetzner.cloud/v1/floating_ips/246612/actions/assign'
```

```
        echo "Master"
```

```
        exit 0
```

```

;;
"BACKUP") echo "backup"
    exit 0
;;
"FAULT") echo "fault"
    exit 0
;;
*)    echo "unknown state"
    exit 1
;;
esac

```

Zurück in der Konfigurationsdatei nach dem Exkurs zum allstateshscript.

Der Virtualserver-Bereich ist in zwei Abschnitte unterteilt. Der Virtualserver selbst und die Realservers. In den Virtualserver-Einstellungen legen wir den Load-Balancer-Modus, die virtuelle IP und den Port fest, auf den der Server hört.

Im Realservers-Abschnitt befinden sich die Checks und die Serveradresse. So haben wir eine Liste von Realservers. Dies sind die Server, die im Failover ausbalanciert werden...

Die Checks können auch benutzerdefinierte Skripte sein. Für OpenVPN verwenden wir ein Python3-Skript von Icinga2, das die Verfügbarkeit von OpenVPN überprüft. Dieses Skript gibt 0 für Erfolg und 1 für Fehlschlag zurück.

Ideal für unseren OpenVPN-Service-Check. Das `openvpn_check`-Skript ist hier als Anhang:

[check\\_vpn.py](#).

Im `check_vpn`-Skript müssen Sie den Rückgabecode bei "crit" von 2 auf 1 ändern.

Sie müssen das Skript ausführbar machen: `chmod +x check_vpn.py`.

```

#====Virtual Server - Definition which floating ip and Ports
#==== Virtual Server for OpenVPN UDP 1194====#
virtual_server 49.12.154.74 1194 { #Virtueller IP-Dienst
    delay_loop 6          #Intervall für die Überprüfung des tatsächlichen Servers festlegen
    lvs_sched fo          #LVS-Zeitplanalgorithmus angeben (fo, wechselt bei Gewicht, das höchste gewinnt)
    #if Dr Mode dont forget to set iptables nat to the virtual ip on the realservers eg. on 49.12.154.75
    #iptables -t nat -A PREROUTING -d 49.12.154.74 -j REDIRECT
    #its virtual ip : 49.12.154.74
    #LVS direct routing
    lvs_method DR          #Specify LVS mode, mainly NAT, TUN, DR
    protocol UDP          #Forwarding protocol is UDP
#Real-Server - here Specify nodes 1-n. Can have an other port, when whished
#==== Realserver UDP 1194 VPN02
real_server 49.12.154.75 1194 { #Backend Real Server Configuration for VPN01
    weight 10            #Set the weight value of the server node. The highest is preferred connection, if

```

available.

```
#this is an Standard TCP check, but we will check UDP VPN we need an own check script, that return 0 or 1.
```

0 = OK, 1 = fail

```
#TCP_CHECK {          #Real Server State Detection Settings section, in seconds
```

```
# connect_timeout 3    #Connection timeout
```

```
# nb_get_retry 3      #retry count
```

```
# delay_before_retry 3 #retry interval
```

```
# connect_port 80     #Connection Port
```

```
#}
```

```
#This done with, misc check
```

```
MISC_CHECK {
```

```
    #misc_path /path/to/check_whatever.sh or py. Hint when parameters given it must be
```

quoted

```
    #misc_path "/root/check_vpn.py -p 1194 --tls-auth /root/ta.key 49.12.154.76" this is when tls key is
```

needed

```
    misc_path "/root/check_vpn.py -p 1194 49.12.154.76"
```

```
    }
```

```
}
```

```
#===== Realserver UDP 1194 LB02 =====#
```

```
real_server 49.12.154.76 1194 { #Backend Real Server Configuration for VPN02
```

```
    weight 9          #Set the weight value of the server node. The highest is preferred connection, if
```

available.

```
#this is an Standard TCP check, but we will check UDP VPN we need an own check script, that return 0 or 1.
```

0 = OK, 1 = fail

```
#TCP_CHECK {          #Real Server State Detection Settings section, in seconds
```

```
# connect_timeout 3    #Connection timeout
```

```
# nb_get_retry 3      #retry count
```

```
# delay_before_retry 3 #retry interval
```

```
# connect_port 80     #Connection Port
```

```
#}
```

```
#This done with, misc check
```

```
MISC_CHECK {
```

```
    #misc_path /path/to/check_whatever.sh or py. Hint when parameters given it must be quoted
```

```
    #misc_path "/root/check_vpn.py -p 1194 --tls-auth /root/ta.key 49.12.154.76" this is when tls key is
```

needed

```
    misc_path "/root/check_vpn.py -p 1194 49.12.154.76"
```

```
    }
```

```
}
```

```

#==== End Realsserver =====#

}

#=== End Virtual Server UDP 1194=====#

```

Der nächste Virtual Server HTTPS 443, hier ist auch vpn auf 443 das sogenannte split tunneling, wird per webbrowser angefragt gehts auf einen webserver, aber wir wollen hier den VPN dienst prüfen

```

#===Virtual Server TCP VPN 443
virtual_server 49.12.154.74 443 { #Virtual IP Service
    delay_loop 6          #Set interval to check actual server
    lvs_sched fo          #Specify LVS Scheduling Algorithm (fo, switched at weight, the heighest wins)
    #f Dr Mode dont forget to set iptables nat to the virtual ip on the realservers eg. on 49.12.154.75
    #iptables -t nat -A PREROUTING -d 192.168.35.127 -j REDIRECT
    #its virtual ip : 49.12.154.74
    lvs_method DR        #Specify LVS mode, mainly NAT, TUN, DR
#   persistence_timeout 50    #Session Hold Time
    protocol TCP         #Forwarding protocol is TCP
#Real-Server - here Specify nodes 1-n. Can have an other port, when whished
    #==== Realsserver TCP 443 VPN02
    real_server 49.12.154.75 443 { #Backend Real Server Configuration for VPN01
        weight 10          #Set the weight value of the server node. The highest is preferred connection, if
available.
        MISC_CHECK {
            #misc_path /path/to/check_whatever.sh or py. Hint when parameters given it must be quoted
            #misc_path "/root/check_vpn.py -p 443 -t --tls-auth /root/ta.key 49.12.154.75" this is when tls key is
needed
            misc_path "/root/check_vpn.py -p 443 -t 49.12.154.75"
        }
    }
    #==== Realsserver TCP 443 LB02 =====#
    real_server 49.12.154.76 443 { #Backend Real Server Configuration for VPN02
        weight 9          #Set the weight value of the server node. The highest is preferred connection, if
available.
        MISC_CHECK {
            #misc_path /path/to/check_whatever.sh or py. Hint when parameters given it must be quoted
            #misc_path "/root/check_vpn.py -p 443 -t --tls-auth /root/ta.key 49.12.154.76" this is when tls key is
needed
            misc_path "/root/check_vpn.py -p 443 -t 49.12.154.76"

```



```

connect_timeout 3    #Connection timeout
nb_get_retry 3      #retry count
delay_before_retry 3 #retry interval
connect_port 10443  #Connection Port
}

}
#===== End Real Server =====#
}
#===== End Virtual Server TCP 10443
#===== End Virtual Server - Definition which floating ip and Ports =====#
#===== End config file =====#

```

Somit hätten wir die Szenarien einmal mit benutzerdefinierten check und TCP check abgedeckt.

Jetzt ist die Konfigurationsdatei für lb01 fertig.

## Nun Server 2

Wir können diese Konfigurationsdatei kopieren und müssen nur wenige Dinge ändern.  
Globaler Abschnitt.

Hier ändern wir die id auf backup

```

#=====Global Config=====#

....
#Rounting settings #here master on the seconday lb02 it must called be slave
router_id backup #An identity that identifies the keepalived server running (type an string)
}

#===== End Globals =====#

```

Nun die HA vrrp\_instance section

Hier muss der state auf BACKUP geändert werden.

Die Priority von 100 auf 99.

Sollte die Netzwerkkarte eine andere sein, diese natürlich auch anpassen

```

#===== HA for Floating IP ====#
vrrp_instance VI_1 {      #VRRP instance definition section

```

```

state BACKUP          #Specify that the node is the primary node (uppercase) and the standby node is
BACKUP
interface ens18       #Netzwerkschnittstelle, über die die Interne kommunikation laufen soll, heartbeat.
Muss ein Privates netzwerk sein, da hier einj Broadcast gesendet wird.

virtual_router_id 51   #VRRP group name, two nodes need to be set the same to indicate that each node
belongs to the same VRRP group. Must be an integer
priority 99           #Priority of the primary node (1-254), default 100, note that the secondary node priority
needs to be lower than the primary node
advert_int 1          #Set the time interval between synchronization checks between two nodes, the two
nodes need to be consistent. Value in seconds
authentication {      #Set validation information, two nodes need to be consistent
    auth_type PASS     #Set up authentication types, mainly PASS and AH
    auth_pass 8nzlTBS0rjP0i77TgL!    #Set the authentication password, two nodes must use the same
secret under a vrrp_instance

}
virtual_ipaddress {   #Specify virtual IP (floating IPs), two nodes need to be set the same, can have more
than one, one per line. as parameter dev the network device were the ip additionally assigned
    49.12.154.74 dev ens18 #it could be an other interface name on lb02
}

....
#=== End Ha for Floating IP =====#

```

Fertig mit der Konfiguration.

## Kopieren Sie die Konfigurationsdateien und starten der Dienste

nach /etc/keepalived und die check\_vpn.py nach /root.

Hier sind die kompletten Konfigurationsdateien:

[lb01.conf](#)

[lb02.conf](#)

[check\\_vpn.py](#).

Dann auf den Loadbalancern den keepalived service starten

```
service keepalived restart
```

Nun noch auf den Servern die route zur floating ip legen

```
/usr/sbin/iptables -t nat -A PREROUTING -d 49.12.154.74 -j REDIRECT
```

Dieses kann man auch in die interfaces als post-up Befehl integrieren.

Beispiel:

```
auto enp6s18
iface enp6s18 inet static
    address 192.168.178.138
    netmask 255.255.255.0
    gateway 192.168.178.1
    dns-nameservers 8.8.8.8 8.8.4.4
    post-up /usr/sbin/iptables -t nat -A PREROUTING -d 49.12.154.74 -j REDIRECT
```

Testen Sie die Konfiguration.

Wenn der Service gestartet ist, können wir den Status damit anzeigen. Geben Sie auf beiden Loadbalancern ein:

Wir können auch sehen, welchen Status die Überprüfungen hatten.

Befehl:

```
service keepalived status
```

Ausgabe:

Hier sehen wir das der Dienst HTTPS 10443 nicht mehr auf der 76 verfügbar ist und schwenkt zu 75.

Hier ist ein Server ausgefallen, das ist kein Schwenkt des Loadbalancers

```
service keepalived status
Output from lb01
root@lb01:~# service keepalived status
● keepalived.service - Keepalive Daemon (LVS and VRRP)
   Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2022-03-16 11:29:36 CET; 21s ago
 Main PID: 2900 (keepalived)
    Tasks: 3 (limit: 2340)
   Memory: 2.0M
     CPU: 634ms
   CGroup: /system.slice/keepalived.service
           └─2900 /usr/sbin/keepalived --dont-fork
```

```
|─2902 /usr/sbin/keepalived --dont-fork
```

```
└─2903 /usr/sbin/keepalived --dont-fork
```

```
Mär 16 11:29:40 lb01 Keepalived_healthcheckers[2902]: TCP_CHECK on service [49.12.154.76]:tcp:10443 failed.
```

```
Mär 16 11:29:40 lb01 Keepalived_healthcheckers[2902]: Removing service [49.12.154.76]:tcp:10443 to VS  
[49.12.154.74]:tcp:10443
```

```
Mär 16 11:29:40 lb01 Keepalived_vrrp[2903]: (VI_1) received lower priority (99) advert from 49.12.154.78 -  
discarding
```

```
Mär 16 11:29:40 lb01 Keepalived_vrrp[2903]: (VI_1) Entering MASTER STATE
```

```
Mär 16 11:29:41 lb01 Keepalived_healthcheckers[2902]: TCP connection to [49.12.154.75]:tcp:10443 success.
```

```
Mär 16 11:29:42 lb01 Keepalived_healthcheckers[2902]: TCP connection to [49.12.154.75]:tcp:443 success.
```

```
Mär 16 11:29:42 lb01 Keepalived_healthcheckers[2902]: Misc check for [[49.12.154.76]:udp:1194 VS  
[49.12.154.74]:udp:1194] by [/root/check_vpn.py] failed with retry disabled (exited with status 1).
```

```
Mär 16 11:29:42 lb01 Keepalived_healthcheckers[2902]: Removing service [49.12.154.76]:udp:1194 to VS  
[49.12.154.74]:udp:1194
```

```
Mär 16 11:29:42 lb01 Keepalived_healthcheckers[2902]: TCP_CHECK on service [49.12.154.76]:tcp:443 failed.
```

```
Mär 16 11:29:42 lb01 Keepalived_healthcheckers[2902]: Removing service [49.12.154.76]:tcp:443 to VS  
[49.12.154.74]:tcp:443
```

In this lines we see that the RVS tcp check 76 is not available on 10443 and is removed

```
Mär 16 11:29:40 lb01 Keepalived_healthcheckers[2902]: TCP_CHECK on service [49.12.154.76]:tcp:10443 failed.
```

```
Mär 16 11:29:40 lb01 Keepalived_healthcheckers[2902]: Removing service [49.12.154.76]:tcp:10443 to VS  
[49.12.154.74]:tcp:10443
```

In this lines we see that RVS 76 misc check openvpn fails and is removed

```
Mär 16 11:29:42 lb01 Keepalived_healthcheckers[2902]: Misc check for [[49.12.154.76]:udp:1194 VS  
[49.12.154.74]:udp:1194] by [/root/check_vpn.py] failed with retry disabled (exited with status 1).
```

```
Mär 16 11:29:42 lb01 Keepalived_healthcheckers[2902]: Removing service [49.12.154.76]:udp:1194 to VS  
[49.12.154.74]:udp:1194
```

In this lines we see that RVS 76 tcp check is not available on 443 and is removed

```
Mär 16 11:29:42 lb01 Keepalived_healthcheckers[2902]: TCP_CHECK on service [49.12.154.76]:tcp:443 failed.
```

```
Mär 16 11:29:42 lb01 Keepalived_healthcheckers[2902]: Removing service [49.12.154.76]:tcp:443 to VS  
[49.12.154.74]:tcp:443
```

## Anzeigen des connection status

```
ipvsadm -Ln --stats
```

Ausgabe:

```

root@lb01:~# ipvsadm -Ln --stats
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port          Conns  InPkts  OutPkts  InBytes  OutBytes
-> RemoteAddress:Port
TCP 49.12.154.74:443             0      0      0      0      0
-> 49.12.154.75:443             0      0      0      0      0
TCP 49.12.154.74:10443          0      0      0      0      0
-> 49.12.154.75:10443          0      0      0      0      0
UDP 49.12.154.74:1194           0      0      0      0      0
-> 49.12.154.75:1194           0      0      0      0      0
root@lb01:~#

```

#Hier können wir sehen, dass die virtuelle IP 74 nur auf einen Server, nämlich 75, zeigt, weil die Überprüfungen auf 76 fehlgeschlagen sind.

#Das ist klar, denn wir haben das LAN-Kabel von 76 zu Testzwecken herausgezogen.

Jetzt stecken wir das Kabel wieder ein.

Im Service-Status sehen wir, dass der Service-Status auf 76 automatisch zurückkehrt.

```

root@lb01:~# service keepalived status
● keepalived.service - Keepalive Daemon (LVS and VRRP)
  Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2022-03-16 13:56:27 CET; 17min ago
  Main PID: 13902 (keepalived)
  Tasks: 3 (limit: 2340)
  Memory: 2.0M
  CPU: 29.825s
  CGroup: /system.slice/keepalived.service
          └─13902 /usr/sbin/keepalived --dont-fork
          └─13903 /usr/sbin/keepalived --dont-fork
          └─13904 /usr/sbin/keepalived --dont-fork

Mär 16 14:06:03 lb01 Keepalived_healthcheckers[13903]: Lost quorum 1-0=1 > 0 for VS [49.12.154.74]:tcp:443
Mär 16 14:06:30 lb01 Keepalived_healthcheckers[13903]: TCP connection to [49.12.154.75]:tcp:443 success.
Mär 16 14:06:30 lb01 Keepalived_healthcheckers[13903]: Adding service [49.12.154.75]:tcp:443 to VS
[49.12.154.74]:tcp:443
Mär 16 14:06:30 lb01 Keepalived_healthcheckers[13903]: Gained quorum 1+0=1 <= 10 for VS
[49.12.154.74]:tcp:443
Mär 16 14:07:44 lb01 Keepalived_healthcheckers[13903]: TCP connection to [49.12.154.76]:tcp:10443 success.
Mär 16 14:07:44 lb01 Keepalived_healthcheckers[13903]: Adding service [49.12.154.76]:tcp:10443 to VS

```

```
[49.12.154.74]:tcp:10443
```

```
Mär 16 14:07:44 lb01 Keepalived_healthcheckers[13903]: TCP connection to [49.12.154.76]:tcp:443 success.
```

```
Mär 16 14:07:44 lb01 Keepalived_healthcheckers[13903]: Adding service [49.12.154.76]:tcp:443 to VS
```

```
[49.12.154.74]:tcp:443
```

```
Mär 16 14:12:51 lb01 Keepalived_healthcheckers[13903]: Misc check for [[49.12.154.76]:udp:1194 VS
```

```
[49.12.154.74]:udp:1194] by [/root/check_vpn.py] succeeded.
```

```
Mär 16 14:12:51 lb01 Keepalived_healthcheckers[13903]: Adding service [49.12.154.76]:udp:1194 to VS
```

```
[49.12.154.74]:udp:1194
```

Das ganze nochmlas in

```
ipvsadm -Ln --stats
```

Ausgabe:

```
root@lb01:~# ipvsadm -Ln --stats
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port          Conns  InPkts  OutPkts  InBytes  OutBytes
-> RemoteAddress:Port
TCP 49.12.154.74:443              0      0        0         0         0
-> 49.12.154.75:443              0      0        0         0         0
-> 49.12.154.76:443              0      0        0         0         0
TCP 49.12.154.74:10443           20     148        0     8880         0
-> 49.12.154.75:10443           20     148        0     8880         0
-> 49.12.154.76:10443           0      0         0         0         0
UDP 49.12.154.74:1194            0      0         0         0         0
-> 49.12.154.75:1194            0      0         0         0         0
-> 49.12.154.76:1194            0      0         0         0         0
```

Was passiert, wenn lb01 ausfällt? Zu Testzwecken haben wir den Keepalive-Service auf lb01 gestoppt.

```
root@lb01:~# service keepalived stop
root@lb01:~# service keepalived status
● keepalived.service - Keepalive Daemon (LVS and VRRP)
   Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Wed 2022-03-16 15:38:51 CET; 32min ago
     Process: 16485 ExecStart=/usr/sbin/keepalived --dont-fork $DAEMON_ARGS (code=exited,
status=0/SUCCESS)
    Main PID: 16485 (code=exited, status=0/SUCCESS)
```

CPU: 4min 4.232s

```
Mär 16 15:15:23 lb01 Keepalived_healthcheckers[16486]: TCP connection to [49.12.154.76]:tcp:10443 success.
Mär 16 15:15:23 lb01 Keepalived_healthcheckers[16486]: Adding service [49.12.154.76]:tcp:10443 to VS
[49.12.154.74]:tcp:10443
Mär 16 15:38:50 lb01 Keepalived[16485]: Stopping
Mär 16 15:38:50 lb01 Keepalived_vrrp[16487]: (VI_1) sent 0 priority
Mär 16 15:38:50 lb01 systemd[1]: Stopping Keepalive Daemon (LVS and VRRP)...
Mär 16 15:38:51 lb01 Keepalived_vrrp[16487]: Stopped
Mär 16 15:38:51 lb01 Keepalived[16485]: Stopped Keepalived v2.1.5 (07/13,2020)
Mär 16 15:38:51 lb01 systemd[1]: keepalived.service: Succeeded.
Mär 16 15:38:51 lb01 systemd[1]: Stopped Keepalive Daemon (LVS and VRRP).
Mär 16 15:38:51 lb01 systemd[1]: keepalived.service: Consumed 4min 4.232s CPU time.
root@lb01:~#
```

On lb02

```
root@lb02:~# service keepalived status
```

● keepalived.service - Keepalive Daemon (LVS and VRRP)

Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)

Active: active (running) since Wed 2022-03-16 14:32:29 CET; 1h 40min ago

Main PID: 4733 (keepalived)

Tasks: 3 (limit: 2340)

Memory: 7.4M

CPU: 6min 4.240s

CGroup: /system.slice/keepalived.service

├─4733 /usr/sbin/keepalived --dont-fork

├─4734 /usr/sbin/keepalived --dont-fork

└─4735 /usr/sbin/keepalived --dont-fork

```
Mär 16 15:15:21 lb02 Keepalived_healthcheckers[4734]: Misc check for [[49.12.154.76]:udp:1194 VS
[49.12.154.74]:udp:1194] by [/root/check_vpn.py] succeeded.
Mär 16 15:15:21 lb02 Keepalived_healthcheckers[4734]: Adding service [49.12.154.76]:udp:1194 to VS
[49.12.154.74]:udp:1194
Mär 16 15:15:23 lb02 Keepalived_healthcheckers[4734]: TCP connection to [49.12.154.76]:tcp:10443 success.
Mär 16 15:15:23 lb02 Keepalived_healthcheckers[4734]: Adding service [49.12.154.76]:tcp:10443 to VS
[49.12.154.74]:tcp:10443
Mär 16 15:15:24 lb02 Keepalived_healthcheckers[4734]: Misc check for [[49.12.154.76]:tcp:443 VS
[49.12.154.74]:tcp:443] by [/root/check_vpn.py] succeeded.
Mär 16 15:15:24 lb02 Keepalived_healthcheckers[4734]: Adding service [49.12.154.76]:tcp:443 to VS
```

[49.12.154.74]:tcp:443

Mär 16 15:38:50 lb02 Keepalived\_vrrp[4735]: (VI\_1) Backup received priority 0 advertisement

Mär 16 15:38:51 lb02 Keepalived\_vrrp[4735]: (VI\_1) Entering MASTER STATE

In the last Line we can see he will become master

Now we start on the lb01 again an the status of lb02

```
root@lb02:~# service keepalived status
```

● keepalived.service - Keepalive Daemon (LVS and VRRP)

Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)

Active: active (running) since Wed 2022-03-16 14:32:29 CET; 1h 40min ago

Main PID: 4733 (keepalived)

Tasks: 3 (limit: 2340)

Memory: 7.4M

CPU: 6min 4.240s

CGroup: /system.slice/keepalived.service

├─4733 /usr/sbin/keepalived --dont-fork

├─4734 /usr/sbin/keepalived --dont-fork

└─4735 /usr/sbin/keepalived --dont-fork

Mär 16 15:15:21 lb02 Keepalived\_healthcheckers[4734]: Misc check for [[49.12.154.76]:udp:1194 VS [49.12.154.74]:udp:1194] by [/root/check\_vpn.py] succeeded.

Mär 16 15:15:21 lb02 Keepalived\_healthcheckers[4734]: Adding service [49.12.154.76]:udp:1194 to VS [49.12.154.74]:udp:1194

Mär 16 15:15:23 lb02 Keepalived\_healthcheckers[4734]: TCP connection to [49.12.154.76]:tcp:10443 success.

Mär 16 15:15:23 lb02 Keepalived\_healthcheckers[4734]: Adding service [49.12.154.76]:tcp:10443 to VS [49.12.154.74]:tcp:10443

Mär 16 15:15:24 lb02 Keepalived\_healthcheckers[4734]: Misc check for [[49.12.154.76]:tcp:443 VS [49.12.154.74]:tcp:443] by [/root/check\_vpn.py] succeeded.

Mär 16 15:15:24 lb02 Keepalived\_healthcheckers[4734]: Adding service [49.12.154.76]:tcp:443 to VS [49.12.154.74]:tcp:443

Mär 16 15:38:50 lb02 Keepalived\_vrrp[4735]: (VI\_1) Backup received priority 0 advertisement

Mär 16 15:38:51 lb02 Keepalived\_vrrp[4735]: (VI\_1) Entering MASTER STATE

Mär 16 16:12:21 lb02 Keepalived\_vrrp[4735]: (VI\_1) Master received advert from 49.12.154.77 with higher priority 100, ours 99

Mär 16 16:12:21 lb02 Keepalived\_vrrp[4735]: (VI\_1) Entering BACKUP STATE

```
root@lb02:~#
```

In the last Line we can see he will become backup state again

on the lb01 service status

```
root@lb01:~# service keepalived status
```

```
● keepalived.service - Keepalive Daemon (LVS and VRRP)
```

```
Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)
```

```
Active: active (running) since Wed 2022-03-16 16:12:18 CET; 10s ago
```

```
Main PID: 23921 (keepalived)
```

```
Tasks: 3 (limit: 2340)
```

```
Memory: 2.0M
```

```
CPU: 586ms
```

```
CGroup: /system.slice/keepalived.service
```

```
├─23921 /usr/sbin/keepalived --dont-fork
```

```
├─23922 /usr/sbin/keepalived --dont-fork
```

```
└─23923 /usr/sbin/keepalived --dont-fork
```

```
Mär 16 16:12:19 lb01 Keepalived_healthcheckers[23922]: TCP connection to [49.12.154.75]:tcp:10443 success.
```

```
Mär 16 16:12:19 lb01 Keepalived_vrrp[23923]: (VI_1) received lower priority (99) advert from 49.12.154.78 - discarding
```

```
Mär 16 16:12:20 lb01 Keepalived_vrrp[23923]: (VI_1) received lower priority (99) advert from 49.12.154.78 - discarding
```

```
Mär 16 16:12:21 lb01 Keepalived_healthcheckers[23922]: Misc check for [[49.12.154.76]:udp:1194 VS [49.12.154.74]:udp:1194] by [/root/check_vpn.py] succeeded.
```

```
Mär 16 16:12:21 lb01 Keepalived_vrrp[23923]: (VI_1) received lower priority (99) advert from 49.12.154.78 - discarding
```

```
Mär 16 16:12:21 lb01 Keepalived_vrrp[23923]: (VI_1) Entering MASTER STATE
```

```
Mär 16 16:12:22 lb01 Keepalived_healthcheckers[23922]: Misc check for [[49.12.154.75]:udp:1194 VS [49.12.154.74]:udp:1194] by [/root/check_vpn.py] succeeded.
```

```
Mär 16 16:12:22 lb01 Keepalived_healthcheckers[23922]: Misc check for [[49.12.154.76]:tcp:443 VS [49.12.154.74]:tcp:443] by [/root/check_vpn.py] succeeded.
```

```
Mär 16 16:12:23 lb01 Keepalived_healthcheckers[23922]: Misc check for [[49.12.154.75]:tcp:443 VS [49.12.154.74]:tcp:443] by [/root/check_vpn.py] succeeded.
```

```
Mär 16 16:12:25 lb01 Keepalived_healthcheckers[23922]: TCP connection to [49.12.154.76]:tcp:10443 success.
```

```
root@lb01:~#
```

In this Line

```
....
```

```
Mär 16 16:12:21 lb01 Keepalived_vrrp[23923]: (VI_1) Entering MASTER STATE
```

```
....
```

He will become master again.

Mission accomplished, HA keepalived works!

# Unicast wenn das Heartbeat Netz nicht in einem Netz mit Multicast funktionen ist.

Multicast funktioniert nicht über VPN und nicht im Hetzner internen Netz.

Für Unicast sollte allerdings kein Öffentliches Netz gewählt werden, denn die Authentifizierung wurde in Version 3 entfernt.

Daher nur Private / Dedizierte Verbindungen für Heartbeate verwenden.

Um Unicast zu benutzen in der VVRP einfach folgendes hinzufügen.

Die eigen IP von Privaten / VPN Netz und die IP Adresse vom Privaten / VPN Netz des anderen. Es können auch mehrer einträge gültig sein.

Einmal mit zwei LBs

10.1.0.3 ist die IP von Knoten selbst, hier lb02

10.1.0.2 ist die ip von lb01

...

priority 99

advert\_int 1

unicast\_src\_ip 10.1.0.3

unicast\_peer {

10.1.0.2

}

virtual\_ipaddress {

...

Einmal mit zwei LBs

10.1.0.3 ist die IP von Knoten selbst, hier lb02

10.1.0.2 ist die ip von lb01

10.1.0.4 ist die ip von lb03

Die ips bei Peer müssen untereinander geschrieben sein.

```
...
priority 99
  advert_int 1

  unicast_src_ip 10.1.0.3
  unicast_peer {
    10.1.0.2
    10.1.0.4
  }

  virtual_ipaddress {
...

```

## lvs\_method und lb\_kind

In der Konfiguration von Linux Virtual Server (LVS) innerhalb von `keepalived` oder ähnlichen Load Balancing-Lösungen sind `lvs_method` und `lb_kind` zwei verschiedene Konfigurationseinstellungen, die verschiedene Aspekte des Load Balancing-Verhaltens steuern.

## lvs\_method (Load Balancing Method)

- **Bedeutung:** `lvs_method` bezieht sich auf die Methode, wie der Traffic an die Backend-Server (Real Server) weitergeleitet wird.
- **Typische Methoden:**
  - **NAT (Network Address Translation):** Der eingehende Traffic wird umgeschrieben, sodass er von der IP des Load Balancers auf die IP des Real Servers umgeleitet wird. Die Antwort des Servers wird ebenfalls umgeschrieben, um vom Load Balancer zu stammen.
  - **DR (Direct Routing):** Der eingehende Traffic wird direkt an die Backend-Server weitergeleitet, ohne die Pakete umzuschreiben. Die Server müssen so konfiguriert sein, dass sie den Traffic, der für die virtuelle IP-Adresse bestimmt ist, akzeptieren und verarbeiten.
  - **TUN (Tunneling):** Der Traffic wird in IP-Tunnel eingekapselt und an die Backend-Server gesendet.

# lb\_kind (Load Balancer Kind)

- **Bedeutung:** `lb_kind` definiert, wie der Load Balancer selbst im Netzwerk funktioniert.
- **Typische Arten:**
  - **NAT:** Der Load Balancer führt eine Netzwerkadressübersetzung durch (wie oben beschrieben).
  - **IP-IP:** Der Load Balancer verwendet IP-IP-Tunneling, wobei die ursprünglichen IP-Pakete in neue IP-Pakete eingekapselt werden.
  - **BRIDGE:** Der Load Balancer arbeitet auf der Datenträgerebene (Layer 2) des OSI-Modells, was bedeutet, dass er wie eine Netzwerkbrücke funktioniert.
- Bei LB kind muss allerdings noch in der sysctl folgender Wert auf 1 gestellt werden

```
net.ipv4.ip_nonlocal_bind = 1
```

## Zusammenfassung

- **lvs\_method** steuert, wie der Traffic an die Backend-Server weitergeleitet wird (z.B. NAT, DR, TUN).
- **lb\_kind** definiert, wie der Load Balancer im Netzwerk arbeitet (z.B. NAT, IP-IP, BRIDGE).

Die Auswahl der richtigen Methode und des Typs hängt von Ihrer Netzwerktopologie, Ihren Performance-Anforderungen und anderen spezifischen Bedürfnissen Ihres Netzwerks ab. Jede Methode und jeder Typ hat seine eigenen Vor- und Nachteile in Bezug auf Komplexität, Durchsatz und Flexibilität.

---

Version #12

Erstellt: 29 Oktober 2023 15:27:32 von Admin

Zuletzt aktualisiert: 10 März 2024 19:00:24 von Admin