How to set up a highly available virtual IP for clusters

This page describes how to enhance the high availability (HA) of the control plane for a LXD cluster, through setting up a virtual IP (VIP) as a single access point.

By exposing cluster members to the network and configuring them as remote servers on a client machine, you can control the cluster over the network. This provides high availability: if one cluster member becomes unavailable, you can access the cluster through another.

You can enhance HA by adding a routing service that uses the Virtual Router Redundancy Protocol (VRRP) to configure a single VIP as the access point for the cluster. While the implementation differs, the concept is similar to a floating IP in cloud platforms.

For more information about HA in LXD clusters, including both the control and data planes, see: High availability.

Use Keepalived

While VRRP is implemented by various tools, **Keepalived** is the most commonly used implementation in Linux environments. The VIP configured with Keepalived is only active on one cluster member at a given time (called the `MASTER`), and Keepalived performs regular checks to reassign the VIP to another member (a `BACKUP`) if the `MASTER` fails to respond.

To install Keepalived, run the following commands on each cluster member:

```
sudo apt update
sudo apt -y install keepalived --no-install-recommends
```

> **Tip**
>
> The `--no-install-recommends` flag prevents `apt` from installing the `ipvsadm` package, which is only needed if you're using the IPVS load balancing feature of Keepalived. If you intend to use this feature, run the above command without `--no-install-recommends`.

The configuration file for Keepalived is typically stored at `/etc/keepalived/keepalived.conf`. You must create a configuration file for each cluster member, with one member set with `state MASTER` and the rest with `state BACKUP`.

Example minimal configuration

Example of a minimal Keepalived configuration for three LXD cluster members (`m1`, `m2`, and `m3`):

`keepalived.conf` on m1

```
vrrp_instance VI_1 {
    state MASTER
    interface enp5s0
    virtual_router_id 41
    priority 200
    advert_int 1
    virtual_ipaddress {
        192.0.2.50/24
    }
}
```

`keepalived.conf` on m2 and m3

```
vrrp_instance VI_1 {
    state BACKUP
    interface enp5s0
    virtual_router_id 41
    priority 100
    advert_int 1
    virtual_ipaddress {
        192.0.2.50/24
    }
}
```

Restart the `keepalived` service on each cluster member after creating or editing its configuration file:

```
sudo systemctl restart keepalived
```

On `m1` (the cluster member designated as `MASTER`), run the following command, using the interface you configured in `keepalived.conf`:

```
ubuntu@m1:~$ ip -br addr show <interface>
```

Confirm that in the output, you can see the VIP as an IP address of the interface.

Test the example configuration

This section describes how to conduct a basic test of the example Keepalived configuration, using the CLI.

First, create a container on each of the cluster members. This can be performed from any of the cluster members, using the `--target` flag. Example:

```
ubuntu@m1:~$ lxc init ubuntu:24.04 c1 --target m1
Creating c1
```

```
ubuntu@m1:~$ lxc init ubuntu:24.04 c2 --target m2
Creating c2
```

```
ubuntu@m1:~$ lxc init ubuntu:24.04 c3 --target m3
Creating c3
```

Run the following command from any cluster member to list the created instances:

```
lxc list
```

Confirm that the containers each exist on a different cluster member.

Next, you need a client LXD server that can access the network used by the cluster for external connectivity. On it, add the VIP as a remote server:

```
ubuntu@my-client:~$ lxc remote add my-cluster 192.0.2.50
```

Then check the list of instances running on the cluster from the client machine. Example:

```
ubuntu@my-client:~$ lxc list my-cluster:
```

The output shown should match what you see when you run `lxc list` on any of the cluster members.

Finally, take the cluster member configured as the Keepalived `MASTER` offline so that it is no longer reachable. This should cause Keepalived to automatically move the VIP to one of the `BACKUP` servers.

From the client server, list the instances running on the cluster once more, using the same command as before. You should see the same list as before, with the exception that the container running on the offline cluster member now displays an `ERROR` state. This confirms that you can still run remote commands on the client, meaning that Keepalived has reassigned the `MASTER` role to another cluster member.

Configuration keys

In this section, we provide brief descriptions of the configuration keys used in this guide. Keep in mind that our example minimal configuration does not include authentication and other settings that might be relevant in production. For full configuration details, refer to the official Keepalived documentation.

`state`:

Only one cluster member can be designated the `MASTER` state. The rest must be set as `BACKUP`.

`interface`

This is the interface that carries the subnet used for client access to the cluster. On clusters using OVN networking, this is the uplink network.

`virtual_router_id`

The `virtual_router_id` must be the same in all cluster members. This assigns the cluster members to the same virtual router.

`priority`

This determines the order in which the VIP is allocated to a cluster member. The `MASTER` must always have a higher priority number than any `BACKUP`. The `BACKUP` servers can use the same number to let Keepalived choose the priority, or you can set a specific priority for each server.

`advert_int`

The `advert_int` key sets the **advertisement internal** (in seconds). The `MASTER` sends VRRP advertisements at this interval to tell `BACKUP` servers that it's online. The `BACKUP` servers assume that the `MASTER` is offline if it stops sending these.

`virtual_ipaddress`

This is the VIP exposed for access to the cluster. Select an unused IP from the subnet used for external access by the cluster. It must be identical on all cluster members.

Load balancing

Using a VIP to route requests to a single cluster member can cause high load on that machine. Keepalived also provides a framework for load balancing, using the Linux Virtual Server (IPVS) kernel module. For details, see the Keepalived documentation.

Alternatively, consider combining Keepalived with an implementation of HAProxy. HAProxy is a reverse proxy that can redirect traffic for both TCP and HTTP protocols, which means that it can handle load balancing both API and UI traffic for LXD clusters.

HAProxy can also support the use of ACME (Automatic Certificate Management Environment) services such as Let's Encrypt to automate renewing certificates for UI access. For details, see: TLS server certificate.

Related topics

How-to guides:

- Clustering

Reference:

- Clusters

Explanation:

- Clusters