Keepalived

What is Keepalived?

Keepalived Basic Explanation

Keepalived is an open-source service commonly used to ensure high availability for network services. It helps by monitoring the health of services and automatically switching over to a backup server if the primary server goes down. This process is known as failover. Keepalived works by using a protocol called VRRP (Virtual Router Redundancy Protocol) to create a virtual IP address that is shared between multiple servers. The virtual IP always points to the currently active (healthy) server, ensuring that users can access services without interruption, even if one server fails.

Keepalived constantly monitors the health of the primary server. If a failure is detected, it assigns the virtual IP address to a backup server, which takes over, keeping the service running smoothly.

Why Use Keepalived for Syncing Docker Swarm Nodes?

In a Docker Swarm environment, you may want to ensure that your services are highly available even if a node goes down. This is where Keepalived becomes useful. By using Keepalived, you can provide a single, consistent virtual IP address that always points to the active node in your cluster. This means that users or external services only need to connect to one IP address, and they will always be directed to a healthy node.

When used on the host system rather than inside Docker, Keepalived can manage failovers at the network level. This ensures that the Docker Swarm services running in replicated mode remain accessible through a single virtual IP, even if the node hosting the service fails. It provides seamless failover without relying on Docker’s internal mechanisms, giving more flexibility and reliability in how the swarm nodes are managed.

By using Keepalived, you make sure that the services running on Docker Swarm remain accessible at all times, providing an extra layer of high availability on top of the replication and scaling features of Docker Swarm.

Simple Explanation of a 3-Node Docker Swarm with Keepalived

In a 3-node Docker Swarm setup, you have three Raspberry Pis (or servers) working together to run your services. Docker Swarm distributes your services across the nodes for high availability and load balancing.

By adding Keepalived to this setup, you can ensure that the cluster is always accessible through a single virtual IP (VIP). In this example, the VIP is `192.168.0.200`. Keepalived monitors the health of the nodes, and if the active node fails, it automatically assigns the VIP to another healthy node.

So, any traffic to `192.168.0.200` is always directed to an available node, ensuring uninterrupted service even if one node goes down.

Step-by-Step Guide: Setup Keepalived

Keepalived on Docker Swarm
A Custom Raspberry Pi Setup

Keepalived is an open-source tool used to ensure high availability and redundancy for services. It does this by monitoring the health of your network and services and automatically switching to a backup server if the primary one fails. In this setup, we are using a custom Keepalived Docker image built specifically for Raspberry Pi (ARMv8) by Takabu, a friend of mine. This allows us to implement Keepalived in a Docker Swarm environment on Raspberry Pis.

> In our Docker Swarm setup, Keepalived is used to provide a Virtual IP (VIP) that will always point to a healthy node, ensuring seamless failover if a node goes offline.

# Priority List and How it Works

Keepalived assigns a priority to each node. The node with the **highest priority** becomes the primary holder of the VIP. If that node goes down, the node with the **next highest priority** takes over the VIP. This ensures that the VIP is always assigned to a functioning node.

Here’s how the priority system works in our Docker Swarm:

- **Swarm1 (192.168.0.10)**: Priority `100` (lowest priority)
- **Swarm2 (192.168.0.11)**: Priority `101`

- **Swarm3 (192.168.0.12)**: Priority `102` (highest priority)

The node with priority 102 (`Swarm3`) will be the primary node, and if it goes down, the VIP will switch to the node with priority 101 (`Swarm2`), and so on.

# Setting Up the Priority List

Before deploying Keepalived, you need to set the priority for each node. Use the following commands to label each node with its priority:

```
docker node ls  # List all nodes in the swarm

# Assign priority labels to each node:
docker node update Swarm1 --label-add KEEPALIVED_PRIORITY=100
docker node update Swarm2 --label-add KEEPALIVED_PRIORITY=101
docker node update Swarm3 --label-add KEEPALIVED_PRIORITY=102
```

> With the priorities set, Keepalived will ensure that the node with the highest priority is always assigned the VIP. If that node fails, Keepalived automatically shifts the VIP to the next highest-priority node, maintaining service availability.

# Docker Compose File for Keepalived

```
version: '3.8'

services:
  keepalived:
    image: takabu/public:docker-swarm-keepalived  # Custom Keepalived image for Raspberry Pi (ARMv8)
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock  # Mount Docker socket to interact with Docker API and control the nodes
    networks:
      - host  # Use the host network for direct access to networking features
    deploy:
      mode: global  # Ensures Keepalived runs on all manager nodes
      placement:
        constraints: [node.role == manager]  # Limit deployment to Swarm manager nodes only
    environment:
      KEEPALIVED_VIRTUAL_IPS: "192.168.0.200"  # Virtual IP (VIP) for Keepalived to manage and switch between nodes

networks:
  host:
    external: true
    name: host  # Leverage the host network to manage VIP switching and network traffic directly
```

Explanation of Key Sections:

- **`/var/run/docker.sock:/var/run/docker.sock`**: This mounts the Docker socket into the Keepalived container, allowing Keepalived to communicate with Docker and control the failover between nodes based on the priority list.
- **`KEEPALIVED_VIRTUAL_IPS: "192.168.0.200"`**: This environment variable specifies the VIP (`192.168.0.200`) that Keepalived will manage. This IP will always point to the highest-priority, healthy node.