**Задание 2.1**

Создать 2 WEB сервера с выводом страницы «Hello Word! \n Server 1» (аналогично для второго Server 2). Сделать балансировку нагрузки (HA + keepalived), чтобы при обновлении страницы мы попадали на любой из WEB серверов(Для балансировки можно сделать 2 отдельных сервера, в сумме 4).

Будет плюсом использование Docker.

Работу стенда продемонстрировать в отчёте.

Решил выполнить с использованием Docker

Воспользовался образами с <https://hub.docker.com/>

docker pull nginx

docker pull haproxy

docker pull osixia/keepalived

создал 4 виртуальных машины и установил на них docker, перенёс конфигурационные файлы и заготовленные index.html. (подготовительные работы выполнил с помощью Vagrant)

2 виртуальные машины (10.10.2.3/24, 10.10.2.4/24) с контейнерами Nginx

sudo docker run -d --net=host --restart unless-stopped --name nginx-1 -v /home/vagrant/nginx/nginx.conf:/etc/nginx/nginx.conf -v /home/vagrant/html:/var/www/html -e TZ=UTC ubuntu/nginx:latest

sudo docker run -d --net=host --restart unless-stopped --name nginx-2 -v /home/vagrant/nginx/nginx.conf:/etc/nginx/nginx.conf -v /home/vagrant/html:/var/www/html -e TZ=UTC ubuntu/nginx:latest

создал 2 файла index.html с разным содержимым для наглядности работы haproxy

set@set-desktop:~$ cat /home/set/work/html-1/index.html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="utf-8">

</head>

<body>

<header>

<h1>Hello Word! <br> Server 1 </h1>

</header>

</body>

</html>

set@set-desktop:~$ cat /home/set/work/html-2/index.html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="utf-8">

</head>

<body>

<header>

<h1>Hello Word! <br> Server 2</h1>

</header>

</body>

</html>

создал конфигурационный файл для веб сервера

set@set-desktop:~$ cat /home/set/work/nginx.conf

user www-data;

worker\_processes auto;

pid /run/nginx.pid;

events { }

http {

server {

listen 80 default\_server;

listen [::]:80 default\_server;

root /var/www/html;

index index.html;

server\_name worktest-site.com;

}

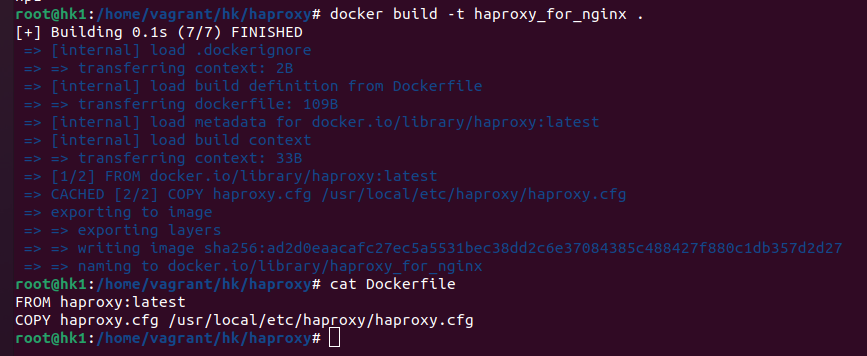
}

далее на 2х других виртуальных машинах (10.10.2.5/24, 10.10.2.6/24) запустил контейнеры keepalived, настройки конфигурации задал переменными при запуске контейнера:

docker run -d --name hk1 --net=host --restart unless-stopped --cap-add=NET\_BROADCAST --cap-add=NET\_RAW --cap-add=NET\_ADMIN --env KEEPALIVED\_UNICAST\_PEERS="#PYTHON2BASH:['10.10.2.5', '10.10.2.6']" --env KEEPALIVED\_INTERFACE="eth1" --env KEEPALIVED\_PRIORITY="110" --env KEEPALIVED\_VIRTUAL\_IPS="10.10.2.10/24" --env KEEPALIVED\_STATE="MASTER" osixia/keepalived:latest

docker run -d --name hk2 --net=host --restart unless-stopped --cap-add=NET\_BROADCAST --cap-add=NET\_RAW --cap-add=NET\_ADMIN --env KEEPALIVED\_UNICAST\_PEERS="#PYTHON2BASH:['10.10.2.5', '10.10.2.6']" --env KEEPALIVED\_INTERFACE="eth1" --env KEEPALIVED\_PRIORITY="100" --env KEEPALIVED\_VIRTUAL\_IPS="10.10.2.10/24" --env KEEPALIVED\_STATE="BACKUP" osixia/keepalived:latest

далее на тех же виртуальных машинах (10.10.2.5/24, 10.10.2.6/24) дополнил контейнер своим конфигом и запустил контейнеры haproxy:



docker run -d --name hp1 -p 8080:80 haproxy\_for\_nginx

docker run -d --name hp2 -p 8080:80 haproxy\_for\_nginx

cat haproxy.cfg

global

log /dev/log local0

log /dev/log local1 notice

stats timeout 30s

daemon

defaults

log global

mode http

option httplog

option dontlognull

timeout connect 5000

timeout client 50000

timeout server 50000

frontend http\_front

bind \*:80

stats uri /haproxy?stats

default\_backend http\_back

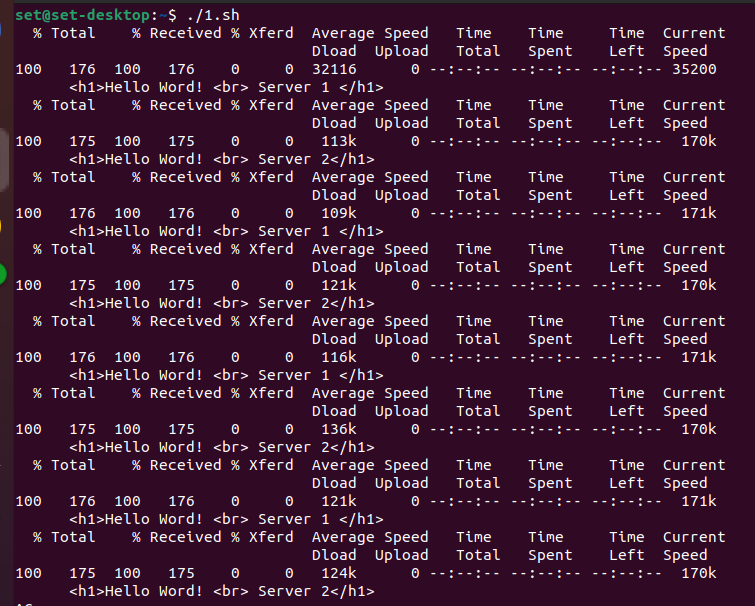
backend http\_back

balance roundrobin

server ng1 10.10.2.3:80 check

server ng2 10.10.2.4:80 check

для проверки haproxy запустил скрипт:

#!/bin/bash

while true

do

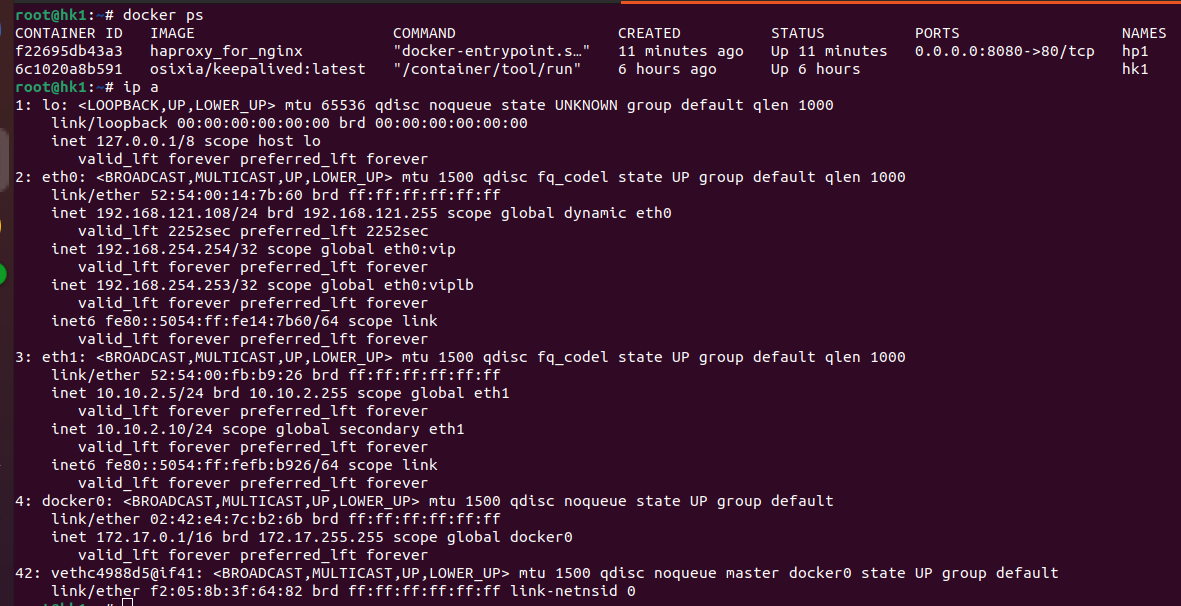
curl worktest-site.com | grep Hello

sleep 1

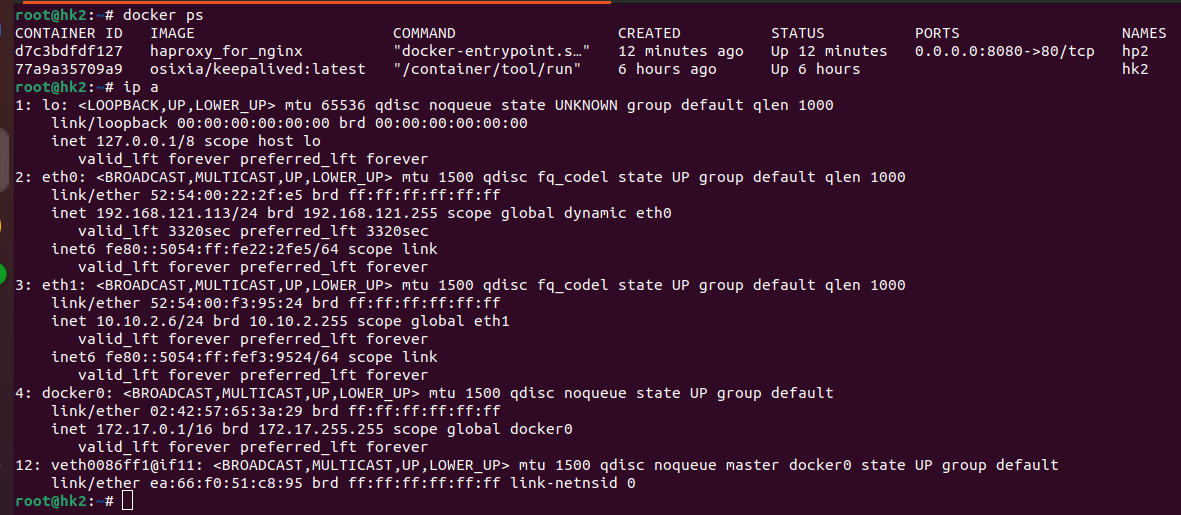
done

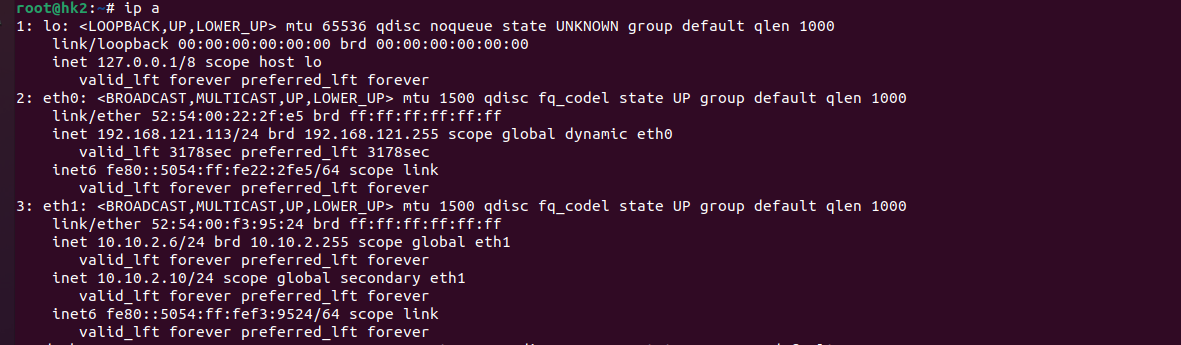
балансировка производиться, это видно по смене наименования сервера.

Далее проверим keepalived, изначально виртуальный ИП находиться на hk1



на hk2 его нет

теперь выключим ВМ hk1 и проверим перешёл ли виртуальный ИП на hk2



VIP действительно перешёл на hk2 балансировка так же продолжает работать.

Задание выполнено.

Задание выложил на github

https://github.com/DimShpInt/worktest