Phân hoạch:

| **Hostname** | **Interface** |
| --- | --- |
| node1 | ens3: 172.16.7.153 (Public) – eth0: 192.168.60.210 |
| node2 | ens3: 172.16.7.163 (Public) – eth0: 192.168.60.220 |
| node3 | ens3: 172.16.7.173 (Public) – eth0: 192.168.60.230 |

Đây là hệ điều hành CentOS

Tạo một server backend đơn giản, lắng nghe ở port 8081 ở cả 3 node

yum install httpd -y

cat /etc/httpd/conf/httpd.conf | grep 'Listen 80'

sed -i "s/Listen 80/Listen 192.168.60.210:8081/g" /etc/httpd/conf/httpd.conf

echo '<h1>Đây là website từ Node 1</h1>' > /var/www/html/index.html

systemctl start httpd

systemctl enable httpd

1. Cài đặt Keepalived

yum install keepalived -y

Cấu hình Keepalived nằm ở: /etc/keepalived/keepalived.conf

Node master:

echo '

vrrp\_script chk\_haproxy {

script "killall -0 haproxy"

interval 2

weight 2

}

vrrp\_instance VI\_1 {

interface ens3

state MASTER

virtual\_router\_id 51

priority 101

virtual\_ipaddress {

172.16.7.200/24

}

track\_script {

chk\_haproxy

}

}' > /etc/keepalived/keepalived.conf

Node 2 và 3 giống nhau, chỉ khác prority:  
echo '

vrrp\_script chk\_haproxy {

script "killall -0 haproxy"

interval 2

weight 2

}

vrrp\_instance VI\_1 {

interface ens3

state BACKUP

virtual\_router\_id 51

priority 100

virtual\_ipaddress {

172.16.7.200/24

}

track\_script {

chk\_haproxy

}

}' > /etc/keepalived/keepalived.conf

Trong đó đối với block vrrp\_script:

* **script “killall -0 haproxy**: kiểm tra tiến trình HAProxy
* **interval 2**: việc kiểm tra được thực hiện mỗi 2 giây
* **weight 2**: nếu kiểm tra thành công thì được +2 điểm

Với block vrrp\_instance:

* **interface ens3**: tên card mạng
* **state MASTER**: chỉ định node này là MASTER hay SLAVE
* **virtual\_router\_id 51**: mỗi cụm Keepalive chỉ cần 1 ID chia sẻ giữa các node, ở đây ta chọn 51
* **priority 101**: giá trị ưu tiên, node nào có priority cao nhất sẽ được chọn để hiển thị web
* **virtual\_ipaddress**: khai báo 1 IP cùng dải với ens3, ví dụ 172.16.7.200
* **track\_script**: giúp Keepalive xác định node nào sẽ nắm IP Virtual. Ví dụ Node 1 có priority là 101 thì Node 1 sẽ nắm IP Virtual. Nếu khi check HA service không thành công thì giá trị này sẽ giảm đi 2 và thấp hơn Node 2, lúc này Node 2 sẽ nắm IP Virtual.

Sửa thêm cấu hình các node để cho phép bind vIP:

echo 'net.ipv4.ip\_nonlocal\_bind = 1' >> /etc/sysctl.conf

echo 'net.ipv4.ip\_forward = 1' >> /etc/sysctl.conf

Start lại dịch vụ: keepalived:

systemctl start keepalived

1. Cài đặt HAProxy

Cài trên cả 3 node từ rpm

Tạo backup cấu hình mặc định:

cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg.bak

echo 'global

log 127.0.0.1 local2

chroot /var/lib/haproxy

pidfile /var/run/haproxy.pid

maxconn 4000

user haproxy

group haproxy

daemon

stats socket /var/lib/haproxy/stats

defaults

mode http

maxconn 8000

log global

option httplog

option dontlognull

option http-server-close

retries 3

timeout http-request 20s

timeout queue 1m

timeout connect 10s

timeout client 1m

timeout server 1m

timeout http-keep-alive 10s

timeout check 10s

listen stats

bind \*:8080 interface ens3

mode http

stats enable

stats uri /stats

stats realm HAProxy\ Statistics

stats admin if TRUE

listen web-backend

bind \*:80

balance roundrobin

cookie SERVERID insert indirect nocache

mode http

option httpchk

option httpclose

option httplog

option forwardfor

server node1 192.168.60.210:8081 check cookie node1 inter 5s fastinter 2s rise 3 fall 3

server node2 192.168.60.220:8081 check cookie node2 inter 5s fastinter 2s rise 3 fall 3

server node3 192.168.60.230:8081 check cookie node3 inter 5s fastinter 2s rise 3 fall 3' > /etc/haproxy/haproxy.cfg

systemctl restart haproxy

systemctl enable haproxy

Giải thích:  
**global Section**

1. **log 127.0.0.1 local2**
   * Specifies that HAProxy will log messages to the local syslog server (127.0.0.1 is the IP address of the syslog server) with the facility local2. This helps in tracking HAProxy events.
2. **chroot /var/lib/haproxy**
   * Sets the chroot (change root) directory for HAProxy. It restricts HAProxy to only access files within the /var/lib/haproxy directory, improving security by isolating it from the rest of the system.
3. **pidfile /var/run/haproxy.pid**
   * Defines the location of the file where HAProxy stores its process ID (PID).
4. **maxconn 4000**
   * Limits the maximum number of simultaneous connections HAProxy will handle to 4000.
5. **user haproxy**
   * Runs HAProxy with the permissions of the haproxy user.
6. **group haproxy**
   * Runs HAProxy with the permissions of the haproxy group.
7. **daemon**
   * Tells HAProxy to run in the background as a daemon.
8. **stats socket /var/lib/haproxy/stats**
   * Enables a Unix socket for accessing HAProxy statistics and configuration. This allows external tools to interact with HAProxy's runtime status.

**defaults Section**

1. **mode http**
   * Sets the default mode to HTTP. This means HAProxy will work with HTTP requests unless explicitly defined otherwise.
2. **maxconn 8000**
   * Limits the maximum number of simultaneous connections per frontend or backend to 8000.
3. **log global**
   * Inherits the global logging configuration (as defined earlier).
4. **option httplog**
   * Logs HTTP requests with details such as the request method, status code, and other HTTP-specific data.
5. **option dontlognull**
   * Prevents logging of requests that result in a null response, i.e., when no request was actually made.
6. **option http-server-close**
   * Instructs HAProxy to close the connection after the HTTP response is sent, rather than keeping the connection alive.
7. **retries 3**
   * Defines the number of retry attempts HAProxy will make before considering a server as unavailable.
8. **timeout http-request 20s**
   * Specifies a 20-second timeout for receiving an HTTP request from the client.
9. **timeout queue 1m**
   * Sets the maximum time a request can stay in the queue before being processed to 1 minute.
10. **timeout connect 10s**
    * Sets the maximum time to establish a connection to a server to 10 seconds.
11. **timeout client 1m**
    * Sets the maximum time to receive data from a client to 1 minute.
12. **timeout server 1m**
    * Sets the maximum time to receive data from a server to 1 minute.
13. **timeout http-keep-alive 10s**
    * Sets the timeout for an HTTP keep-alive connection to 10 seconds.
14. **timeout check 10s**
    * Sets the timeout for checking the health of a backend server to 10 seconds.

**listen stats Section**

1. \**bind :8080 interface ens3*
   * Listens on all interfaces (\*) and binds to port 8080 on interface ens3 for the stats page.
2. **mode http**
   * Operates in HTTP mode, as the stats page is HTTP-based.
3. **stats enable**
   * Enables the statistics page for HAProxy.
4. **stats uri /stats**
   * Specifies the URI to access the stats page (/stats), e.g., http://<haproxy-server>:8080/stats.
5. **stats realm HAProxy\ Statistics**
   * Sets the authentication realm for the stats page, which is useful for configuring HTTP authentication.
6. **stats admin if TRUE**
   * Grants access to the stats page only if the TRUE condition is met. This could be a security check, like requiring the request to come from a specific source or IP.

**listen web-backend Section**

1. \**bind :80*
   * Listens on port 80 on all interfaces to handle HTTP traffic.
2. **balance roundrobin**
   * Uses the round-robin algorithm to balance incoming requests across the backend servers.
3. **cookie SERVERID insert indirect nocache**
   * Inserts a cookie (SERVERID) in the response to track the session and maintain persistence (stickiness) across requests.
     + insert: Add a cookie if not present.
     + indirect: Marks the cookie for HTTP clients.
     + nocache: Prevents caching of the cookie.
4. **mode http**
   * Operates in HTTP mode for handling HTTP requests.
5. **option httpchk**
   * Enables HTTP health checks for the backend servers.
6. **option httpclose**
   * Ensures that the HTTP connection is closed after each request/response cycle.
7. **option httplog**
   * Logs HTTP traffic with detailed request and response information.
8. **option forwardfor**
   * Adds the X-Forwarded-For header to HTTP requests to pass the client’s IP address to backend servers, which is useful for logging and security purposes.
9. **server node1 192.168.60.210:8081 check cookie node1 inter 5s fastinter 2s rise 3 fall 3**
   * Configures a backend server (node1) at IP 192.168.60.210 on port 8081 with the following settings:
     + check: Enables health checks for this server.
     + cookie node1: Uses node1 as the server's unique cookie value for session persistence.
     + inter 5s: Health check interval is 5 seconds.
     + fastinter 2s: Uses a faster check interval of 2 seconds when the server is down.
     + rise 3: The server must pass 3 consecutive health checks to be marked as up.
     + fall 3: The server is considered down after 3 consecutive failed health checks.
10. **server node2 192.168.60.220:8081 check cookie node2 inter 5s fastinter 2s rise 3 fall 3**
    * Same as above for a second server (node2) at IP 192.168.60.220.
11. **server node3 192.168.60.230:8081 check cookie node3 inter 5s fastinter 2s rise 3 fall 3**
    * Same as above for a third server (node3) at IP 192.168.60.230.