HAProxy Lab Setup Guide : Multi-OS Installation

Prerequisites

- 3 VMs (or use VirtualBox/VMware Workstation to create them)
- Web browser access (for those using AFNOG infrastructure)

VM Setup

1. **VM1:** HAProxy
   - IP: 192.168.1.X
2. **VM2:** Apache Server
   - IP: 192.168.1.Y
3. **VM3:** Nginx Server
   - IP: 192.168.1.Z

Local Hosts File Configuration

Add the following entries to your local hosts file, pointing them all to the HAProxy IP (192.168.1.X):

```
192.168.1.X lb.lab.afnog.org
192.168.1.X www.lab.afnog.org
192.168.1.X nginx.lab.afnog.org
192.168.1.X apache.lab.afnog.org
```

Step 1: Install and Configure HAProxy (VM1)

Red Hat-based systems (CentOS, Fedora)

```
sudo yum update
sudo yum install haproxy
```

Debian-based systems (Ubuntu, Debian)

```
sudo apt update
sudo apt install haproxy
```

FreeBSD

```
sudo pkg update
sudo pkg install haproxy
```

Step 2: Install and Configure Apache (VM2)

Red Hat-based systems

```
sudo yum update

sudo yum install httpd

sudo systemctl start httpd

sudo systemctl enable httpd
```

Debian-based systems

```
sudo apt update
sudo apt install apache2
```

FreeBSD

```
sudo pkg update
sudo pkg install apache24
sudo sysrc apache24_enable="YES"
sudo service apache24 start
```

Create a custom index.html:

`echo "This is the Apache Server" | tee /var/www/html/index.html`

On FreeBSD

`echo "This is the Apache Server" | tee /usr/local/www/apache24/data/index.html`

Step 3: Install and Configure Nginx (VM3)

Red Hat-based systems

```
sudo yum update
sudo yum install nginx
sudo systemctl start nginx
sudo systemctl enable nginx
```

Debian-based systems

```
sudo apt update
sudo apt install nginx
```

FreeBSD

```
sudo pkg update
sudo pkg install nginx
sudo sysrc nginx_enable="YES"
sudo service nginx start
```

Create a custom index.html:

```
echo "This is the Nginx Server" |  tee /var/www/html/index.html
 # For FreeBSD:
echo "This is the Nginx Server" | tee /usr/local/www/nginx/index.html
```

HAProxy Configuration

Step 1: Basic Frontend and Backend Setup (Round-Robin)

HAProxy Configuration: Edit the HAProxy configuration file:

- **Red Hat and Debian:** /etc/haproxy/haproxy.cfg
- **FreeBSD:** /usr/local/etc/haproxy.conf

Add the following configuration:

```
global
    log         127.0.0.1:514 local1 info
    chroot      /var/empty
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon

defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000

frontend http-in
    bind *:80
    default_backend www_back

backend www_back
    balance roundrobin
    server nginx_server vm1.log.afnog.org:80 check
    server apache_server vm2.lab.afnog.org:80 check
```

Restart HAProxy:

```
systemctl restart haproxy
```

Step 2: Advanced Configuration with ACLs (Access Control Lists)

Updated HAProxy Configuration:

Modify the existing HAProxy configuration to include the following:

```
frontend http_front
    bind *:80
    acl url_nginx hdr(host) -i nginx.lab.afnog.org
    acl url_apache hdr(host) -i apache.lab.afnog.org
    use_backend nginx_back if url_nginx
    use_backend apache_back if url_apache
    default_backend www_back

backend www_back
    balance roundrobin
    server nginx_server 192.168.1.Z:80 check
    server apache_server 192.168.1.Y:80 check

backend nginx_back
    server nginx_server 192.168.1.Z:80 check

backend apache_back
    server apache_server 192.168.1.Y:80 check
```

To set up an active-passive configuration for your backend node, adjust the existing HAProxy configuration to include the following:

```
backend www_back
    balance roundrobin
    server nginx_server 192.168.1.Z:80 check
    server apache_server 192.168.1.Y:80 check backup
```

this setup will make node apache_server as a passive node and will not recive traffic unless node nginx_server is down

Restart HAProxy:

```
sudo systemctl restart haproxy
```

Step 3: Adding a Status Page

Final HAProxy Configuration:

Add the following configuration for the status page:

```
listen stats
    bind *:8404
    stats enable
    stats uri /
    stats refresh 5s
```

Restart HAProxy:

sudo systemctl restart haproxy

Testing the Status Page:

You can access the status page by navigating to http://192.168.1.X:8404/ in your web browser.

SSL Termination on HAProxy

Generate a Self-Signed Certificate:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/haproxy.key -out /etc/ssl/certs/hapro
```

Combine the Certificate and Key:

```
cat /etc/ssl/certs/haproxy.crt /etc/ssl/private/haproxy.key |  tee /etc/ssl/certs/haproxy.pem
```

Note: For development SSL certificates, you can use the repository at https://github.com/BenMorel/dev-certificates

Update HAProxy Configuration to Use SSL:

Add the following to the `frontend http_front` section:

```
bind *:443 ssl crt /etc/ssl/certs/haproxy.pem
redirect scheme https if !{ ssl_fc }
```

Restart HAProxy:

```
sudo systemctl restart haproxy
```

Example for Layer 4 Load balancing , DB port :

```
frontend mysql
 mode tcp
 bind :3306
 default_backend mysql_servers
```

```
backend mysql_servers
 mode tcp
 balance leastconn
 server s1 192.168.0.10:3306 check
 server s2 192.168.0.11:3306 check
```

Configure Syslog for HAProxy Logging

1. Open the syslog configuration file for editing:

```
vi /etc/syslog.conf
```

1. Add the following lines to configure logging:

```
*.err;kern.warning;auth.notice;mail.crit                /dev/console
local1.*                                                /var/log/haproxy.log
*.notice;authpriv.none;kern.debug;lpr.info;mail.crit;news.err   /var/log/messages
```

1. Create the HAProxy log file:

```
touch /var/log/haproxy.log
```

1. Set the appropriate ownership for the log file:

```
chown haproxy:haproxy /var/log/haproxy.log
```

1. Update the syslogd flags to bind to localhost and run in compatibility mode:

```
sysrc syslogd_flags="-b localhost -C"
```

1. Restart the syslog service to apply changes:

```
service syslogd restart
```

Testing

Using `web browser`:

1. Test round-robin for `www.lab.afnog.org`:
2. Repeat the command several times to see alternating responses from Nginx and Apache.

- Test Nginx backend:

```
 nginx.lab.afnog.org
# This should consistently return the Nginx server response.
```

- Test Apache backend:

```
 apache.lab.afnog.org
# This should consistently return the Apache server response.
```

- Test SSL termination:

```
 https://www.lab.afnog.org
# This should return responses over HTTPS, with round-robin load balancing between Nginx and Apache.
```

Troubleshooting: Common Issues and Solutions

HAProxy not starting:

- Check the configuration file for syntax errors:

```
haproxy -c -f /etc/haproxy/haproxy.cfg
```

- Verify that the ports HAProxy is trying to bind to are not in use by other services.

Backend servers not responding:

  - Ensure that Apache and Nginx are running on their respective VMs.
  - Check firewall rules to allow traffic between HAProxy and backend servers.
  - Verify the IP addresses and ports in the HAProxy configuration.

SSL certificate issues:

  - Double-check the path to the SSL certificate and key in the HAProxy configuration.
  - Ensure the combined PEM file has the correct permissions.

ACLs not working as expected:

  - Verify that your local hosts file is correctly configured.
  - Use `tcpdump` or `wireshark` to inspect the HTTP headers and ensure the correct `Host` header is being sent.

Performance Tuning: Optimizing HAProxy

Increase maximum connections:

  - Adjust the `maxconn` parameter in the `global` section based on your server's capacity.

Enable kernel TCP splicing:

- Add `option tcpka` to the `defaults` section for keep-alive connections.

Use HTTP/2:

- Update your SSL binding to support HTTP/2:

```
bind *:443 ssl crt /etc/ssl/certs/haproxy.pem alpn h2,http/1.1
```

Implement caching:

- Consider adding a caching layer with Varnish in front of HAProxy for static content.

Optimal Configuration Options for Web-Based Frontends

It's crucial to customize the following according to your application's specific requirements.

```
frontend http-in
bind *:80
bind *:443 ssl crt /etc/haproxy/certs/cert.pem no-sslv3
mode http
option httplog
log global

# Redirect HTTP to HTTPS (enforce HTTPS for all traffic)
http-request redirect scheme https code 301 if !{ ssl_fc }

# Set default security headers for responses
# Enforce HSTS for HTTPS (1 year, include subdomains, preload)
http-response set-header Strict-Transport-Security "max-age=31536000; includeSubDomains; preload"

# Clickjacking protection, allow only the same origin to embed this site
http-response set-header X-Frame-Options "SAMEORIGIN"

# XSS filtering enabled in browsers, block if an attack is detected
http-response set-header X-XSS-Protection "1; mode=block"

# Prevent MIME type sniffing (force browser to honor content type declared by the server)
http-response set-header X-Content-Type-Options "nosniff"

# Add Content Security Policy to mitigate XSS and data injection attacks
http-response set-header Content-Security-Policy "default-src 'self'; script-src 'self'; object-src 'none'"

# Disable referrer information leakage when navigating to a different origin
http-response set-header Referrer-Policy "no-referrer-when-downgrade"

# Prevent browsers and proxies from caching sensitive data
http-response set-header Cache-Control "no-store, no-cache, must-revalidate, proxy-revalidate, max-age=0"

# Set secure cookies (only for HTTPS, HttpOnly, and prevent cross-site requests)
```

```
acl secure_cookie hdr_sub(cookie) Secure
http-response set-header Set-Cookie %[res.hdr(Set-Cookie)] if secure_cookie
http-response set-header Set-Cookie Secure; HttpOnly; SameSite=Strict if secure_cookie

# Forward client's original IP in X-Forwarded-For header
http-request add-header X-Forwarded-For %[src]

# Forward the protocol used by the client (HTTP/HTTPS) in X-Forwarded-Proto header
http-request add-header X-Forwarded-Proto https if { ssl_fc }
http-request add-header X-Forwarded-Proto http if !{ ssl_fc }

# Preserve the original Host header
http-request add-header X-Forwarded-Host %[req.hdr(host)]

default_backend servers
```

Security Considerations

1. Regularly update HAProxy and backend servers
2. Implement strong SSL/TLS configurations
3. Use IP whitelisting for the HAProxy stats page
4. Consider implementing Web Application Firewall (WAF) rules in HAProxy
5. Regularly audit your HAProxy configurations and access logs

This guide provides a comprehensive setup process for HAProxy, starting from a basic configuration and progressing to more advanced setups with ACLs, SSL termination, and performance optimization. Always ensure to test thoroughly in a staging environment before applying changes to production systems.

Author

Manhal Mohamed , sdnog team

Revision #2
Created 28 October 2024 15:02:58 by sara
Updated 28 November 2024 13:10:52 by sara