Terminate https connections in HAProxy Using Ansible

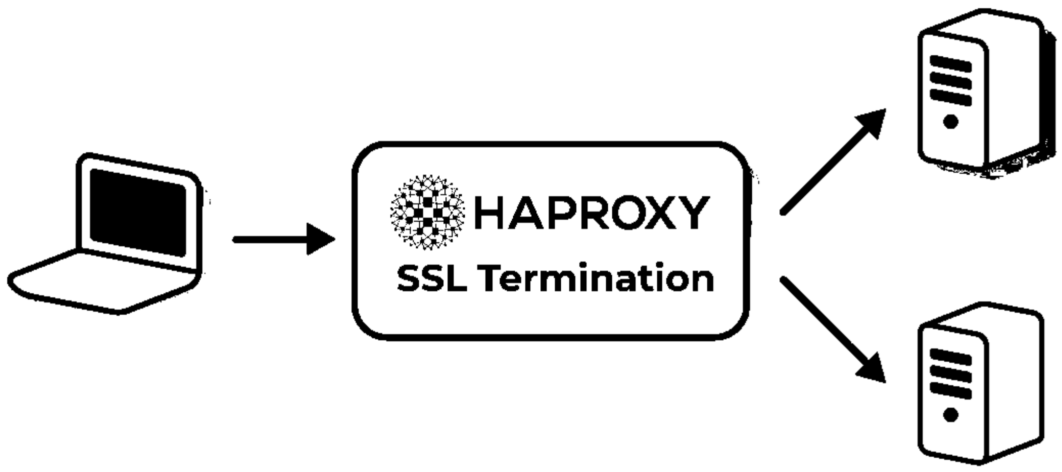
By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

Session - 48 Agenda:

Terminate https connections in HAProxy Using Ansible:

The HAProxy load balancer provides high-performance SSL termination, allowing you to encrypt and decrypt traffic. You can quickly and easily enable SSL/TLS encryption for your applications by using HAProxy SSL termination.

HAProxy is compiled with OpenSSL, which allows it to encrypt and decrypt traffic as it passes. In this blog post, you will learn how to set this up and why delegating this function to HAProxy simplifies your infrastructure.



The Benefits of SSL Termination:

When you operate a farm of servers, it can be a tedious task maintaining SSL certificates. Even using a Let's Encrypt Certbot to automatically update certificates has its challenges because, unless you have the ability to dynamically update DNS records as part of the certificate renewal process, it may necessitate making your web servers directly accessible from the Internet so that Let's Encrypt servers can verify that you own your domain.

Enabling SSL on your web servers also costs more CPU usage, since those servers must become involved in encrypting and decrypting messages. That CPU time could otherwise have been used to do other meaningful work. Web servers can process requests more quickly if they're not also crunching through encryption algorithms simultaneously.

The term SSL termination means that you are performing all encryption and decryption at the edge of your network, such as at the load balancer. The load balancer strips away the encryption and passes the messages in the clear to your servers. You might also hear this called SSL offloading.

SSL termination has many benefits. These include the following:

1. You can maintain certificates in fewer places, making your job easier.
2. You don't need to expose your servers to the Internet for certificate renewal purposes.
3. Servers are unburdened from the task of processing encrypted messages, freeing up CPU time.

Lab Environment:

192.168.229.128 haproxy-server
192.168.229.129 node01
192.168.229.131 node02

Terminate https connections in HAProxy Using Ansible

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

Let's create an Ansible playbook to setup the HAProxy server at the managed node(s).

$ vim https-terminate-haproxy-server.yml

```
---
- name: HAProxy Server Configuration Playbook
  hosts: localhost
  become: true
  tasks:
    - name: Setting up the static hostname in the server machine.
      hostname:
        name: haproxy-server
        use: systemd

    - name: Making entries in the /etc/hosts file for the server hostnames & IP Addresses
      blockinfile:
        dest: /etc/hosts
        block: |
          192.168.229.128 haproxy-server
          192.168.229.129 node01
          192.168.229.131 node02
        insertafter: EOF

    - name: Installing haproxy, openssl & httpd packages in the machine.
      dnf:
        name:
          - haproxy
          - openssl
          - mod_ssl
          - httpd
        state: latest

    - name: Creating /etc/pki/haproxy directory for the SSL private key file.
      file:
        path: /etc/pki/haproxy
        mode: '0755'
        state: directory

    - name: Generating the Private Key File.
      openssl_privatekey:
        path: /etc/pki/haproxy/haproxy.key
        size: 2048

    - name: Generating Certificate Sign Request (CSR)
      community.crypto.x509_certificate:
        path: '/etc/pki/haproxy/haproxy.crt'
        privatekey_path: '/etc/pki/haproxy/haproxy.key'
        force: true
        provider: selfsigned

    - name: Concatenating the key and certificate to a PEM file.
      shell:    cat    /etc/pki/haproxy/haproxy.crt    /etc/pki/haproxy/haproxy.key    >
/etc/pki/haproxy/haproxy.pem
```

```
- name: Copying the /etc/haproxy/haproxy.cfg file using ansible jinja template.
  template:
    src: haproxy.cfg.j2
    dest: /etc/haproxy/haproxy.cfg
    force: true

- name: Making changes in the /etc/rsyslog.conf file.
  replace:
    dest: /etc/rsyslog.conf
    regexp: '^#module(load="imudp")'
    replace: 'module(load="imudp")'

- name: Making changes in the /etc/rsyslog.conf file.
  replace:
    dest: /etc/rsyslog.conf
    regexp: '^#input(type="imudp" port="514")'
    replace: 'input(type="imudp" port="514")'

- name: Creating haproxy.conf file for the rsyslog.
  copy:
   dest: "/etc/rsyslog.d/haproxy.conf"
   content: |
       local2.=info /var/log/haproxy-access.log
       local2.notice /var/log/haproxy-info.log

- name: Restarting & enabling the rsyslog service.
  service:
    name: rsyslog
    state: restarted
    enabled: yes

- name: Turning on the haproxy_connect_any SELinux boolean.
  command: setsebool -P haproxy_connect_any 1

- name: Allowing HTTP & HTTPS traffic in the firewall.
  firewalld:
    service: "{{ item }}"
    zone: public
    permanent: true
    immediate: true
    state: enabled
  loop:
    - https
    - http

- name: Starting & enabling the haproxy service.
  service:
    name: haproxy
    state: started
    enabled: yes
```

Terminate https connections in HAProxy Using Ansible

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

```
- hosts: node1
  become: true
  tasks:
    - name: Setting up the static hostname in the node1 machine.
      hostname:
        name: node01
        use: systemd

    - name: Making entries in the /etc/hosts file for the server hostnames & IP Addresses
      blockinfile:
        dest: /etc/hosts
        block: |
          192.168.229.128 haproxy-server
          192.168.229.129 node01
          192.168.229.131 node02
        insertafter: EOF

    - name: Installing httpd packages in the machine.
      dnf:
        name: httpd
        state: latest

    - name: Copying the image file to the /var/www/html/ directory.
      ansible.builtin.copy:
        src: /home/vikasnehra/NehraClassesLogo.png
        dest: /var/www/html/NehraClassesLogo.png
        mode: '0644'

    - name: Creating the index.html file for node1.
      copy:
        dest: "/var/www/html/index.html"
        content: |
          <img src="/NehraClassesLogo.png" width="500" height="500" />
          <h1>Nehra Classes Are Awesome.</h1>
          <i>This page is hosted on node1 machine using apache.</i>

    - name: Allowing HTTP traffic in the firewall.
      firewalld:
        service: http
        zone: public
        permanent: true
        immediate: true
        state: enabled

    - name: Starting & enabling the HTTPD service.
      service:
        name: httpd
        state: started
        enabled: yes

- hosts: node3
  become: true
```

```
 tasks:
  - name: Setting up the static hostname in the node3 machine.
    hostname:
      name: node02
      use: systemd

  - name: Making entries in the /etc/hosts file for the server hostnames & IP Addresses
    blockinfile:
      dest: /etc/hosts
      block: |
        192.168.229.128 haproxy-server
        192.168.229.129 node01
        192.168.229.131 node02
      insertafter: EOF

  - name: Installing httpd packages in the machine.
    dnf:
      name: httpd
      state: latest

  - name: Copying the image file to the /var/www/html/ directory.
    ansible.builtin.copy:
      src: /home/vikasnehra/NehraClassesLogo.png
      dest: /var/www/html/NehraClassesLogo.png
      mode: '0644'

  - name: Creating the index.html file for node2.
    copy:
      dest: "/var/www/html/index.html"
      content: |
        <img src="/NehraClassesLogo.png" width="500" height="500" />
        <h1>Nehra Classes Are Awesome.</h1>
        <i>This page is hosted on node2 machine using apache.</i>

  - name: Allowing HTTP traffic in the firewall.
    firewalld:
      service: http
      zone: public
      permanent: true
      immediate: true
      state: enabled

  - name: Starting & enabling the httpd service.
    service:
      name: httpd
      state: started
      enabled: yes
...
```

We would require the ansible.posix collection which we can install from Ansible Galaxy.

$ ansible-galaxy collection install ansible.posix

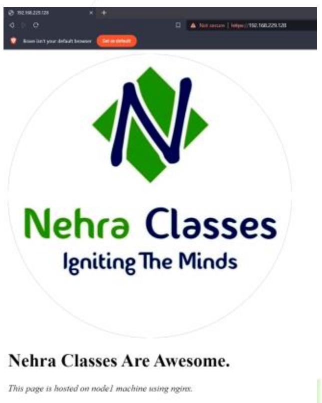
We would also require the community.general collection which we can install from Ansible Galaxy.
$ ansible-galaxy collection install community.general

We would also require the community.crypto collection which we can install from Ansible Galaxy.
$ ansible-galaxy collection install community.crypto

Now, we can execute the ansible playbook to setup the HAProxy server at the managed node(s).
$ ansible-playbook https-terminate-haproxy-server.yml

Login to haproxy server and run the curl command couple of times to see whether traffic is distributed in round-robin way.
$ curl 192.168.229.129
$ curl 192.168.229.131

You can verify the same using any web browser as well. You can see that your traffic is being redirected to https be default, while we didn't create the self-signed certificate on the managed nodes (node01/node02) and didn't allow the https traffic there as well.



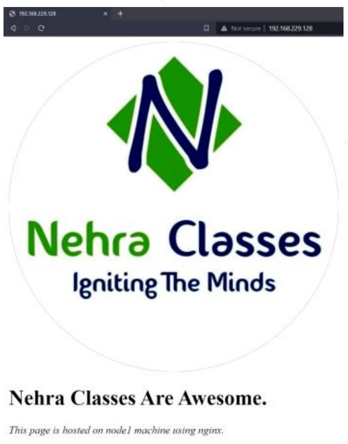
Nehra Classes Are Awesome.

This page is hosted on node1 machine using nginx.

if you want you can comment out the following line in the /etc/haproxy/haproxy.cfg file to cancel this re-direction of the traffic from http to https.

```
# vim /etc/haproxy/haproxy.cfg
# redirect scheme https if !{ ssl_fc }
# systemctl restart haproxy
```

Verify the same from the web browser.



Nehra Classes Are Awesome.

This page is hosted on node1 machine using nginx.

HAProxy server is working in the https connections terminator mode as expected.

Thank You