Session - 43 Agenda:

Apache HTTPS Server Using Ansible:

Here you will learn how to set up or enable Apache HTTPS (SSL) Web server with self-signed SSL certificate on Linux RHEL9 using Ansible. A self-signed certificate will not validate the identity of your server, since it is not signed by a trusted certificate authority, but it will allow you to encrypt communications between your server and your visitors.

Let’s create an ansible playbook to configure Apache HTTPS server at the managed node(s). Here we will create a separate directory under /var/www/html for the virtual hosting.

```
$ vim apache-https.yml
---
- name: Apache HTTPS Server Configuration Playbook
  hosts: node1
  become: true
  tasks:
    - name: Setting up the static hostname in the server machine.
      hostname:
         name: node1.nehraclasses.local
         use: systemd

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

    - name: Generating the Private Key File with the name as localhost.key
      openssl_privatekey:
        path: /etc/pki/tls/private/localhost.key
        size: 2048

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

    - name: Creating /var/www/html/nehraclassesweb/ directory for the document root.
      file:
        path: /var/www/html/nehraclassesweb/
        mode: '0755'
        state: directory

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

    - name: Creating HTTPD configuration file in the /etc/httpd/conf.d/ directory.
      copy:
        dest: "/etc/httpd/conf.d/httpd.conf"
        content: |
          <VirtualHost *:443>
          SSLEngine on
          SSLCertificateFile /etc/pki/tls/certs/localhost.crt
          SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
          ServerName node1.nehraclasses.local
          DocumentRoot /var/www/html/nehraclassesweb/
          </VirtualHost>

    - name: Creating the website index file.
      copy:
        dest: "/var/www/html/nehraclassesweb/index.html"
        content: |
          <h1>Nehra Classes Are Awesome.</h1>
          <i>This page is hosted on node1 machine using apache.</i>
          <img src="/NehraClassesLogo.png" width="300" height="300" />

    - name: Making changes in the /etc/httpd/conf.d/ssl.conf file. (Set the ServerName)
      lineinfile:
        dest: /etc/httpd/conf.d/ssl.conf
        regexp: '^#ServerName'
        line: 'ServerName node1.nehraclasses.local:443'

    - name: Allowing Apache to modify the files in the /var/www/html/nehraclassesweb/ directory. (SELinux Context)
      command: restorecon -Rv /var/www/html/

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

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

Before we execute this playbook, we would require the following collections which need to be installed in the Ansible server.

```
$ ansible-galaxy collection install ansible.posix
$ ansible-galaxy collection install community.general
$ ansible-galaxy collection install community.crypto
```

We would require the python3-pip packages also, so that we can install the cryptography packages using pip command later.

```
$ sudo dnf install -y python3-pip-21.2.3-6.el9.noarch
```

cryptography is a package which provides cryptographic recipes and primitives to Python developers. Our goal is for it to be your “cryptographic standard library”.

You can install cryptography using pip command.

```
$ pip install cryptography==37.0.0
```

We would require the python3-pip packages on the managed node as well, so that we can install the cryptography packages using pip command later there. We can do it using ansible ad-hoc command (or we could have mentioned the same in the playbook as well).

```
$ ansible node1 -m command -a 'sudo dnf install -y python3-pip-21.2.3-6.el9.noarch'
```

Now, you can install cryptography using pip command in form of Ansible ad-hoc command (use sudo for the root).

```
$ ansible node1 -m command -a 'sudo pip install cryptography==37.0.0'
```

Now, we can execute the playbook to setup the Apache HTTPS server on the managed node1.

```
$ ansible-playbook apache-https.yml
```

Now, you can open any web browser and mention the IP Address (hostname if you have an authoritative DNS in your environment) with HTTPS.

https://192.168.229.129/

You can see the web page hosted on the manage node1.

Thank You