CST8246 - Web Server Configuration with Automation

Objectives

- Install and configure an Apache web server.
- Automate the installation and configuration process.
- Set up and manage virtual hosts in Apache.
- Secure the Apache web server using SSL/TLS.
- Verify running services and their listening ports.
- Identify and troubleshoot service issues using log files.

Lab Outcomes

- Successfully install Apache with a default configuration.
- Configure Apache to serve two additional (virtual) websites.
- Deploy an Apache server secured with SSL/TLS.
- Automate installation and configuration using scripts.
- Understand cross-platform differences in Apache setup.
- Verify active services and troubleshoot using logs.

Lab Deliverables

- **Apache installed and accessible** via networkID.exampleMN.lab
- **Two virtual web sites configured** (www.exampleMN.lab, www.siteMN.lab)
- **Apache with SSL/TLS secured** (secure.exampleMN.lab)
- **Automation scripts** for installation and configuration
- **Verification scripts** to confirm proper setup

Section A - Apache Installation and Configuration

1. Installing Apache

- Install or update the **httpd** package and confirm that the installed version is **2.4**.
- Use the following Apache commands for troubleshooting and configuration:

Command	Description
httpd -d directory	Sets a custom server root directory (ServerRoot).
httpd -f filename	Specifies an alternate configuration file.
httpd -V	Displays compiled settings.
httpd -v	Shows the Apache version number.
httpd -l	Lists compiled modules.
httpd -M	Displays all loaded modules (both static and compiled).
httpd -t	Runs a syntax check on configuration files.
httpd -S	Displays virtual host configurations.

- Start the web server using `systemctl`.
- Verify that Apache is running and listening on port **80**.
- Monitor Apache error logs in real-time using:
  - `/etc/httpd/logs/error_log`
  - `journalctl -u httpd -f`
- Test the default web page by opening **localhost** in a browser.

2. Configuring Apache

- Create a new Apache configuration file with the necessary directives.
- Before making changes, **backup** the existing configuration file.

Basic Apache Configuration File:

```
# Define the server name
ServerName hostname.exampleMN.lab
```

```
# Define server root directory
ServerRoot /etc/httpd

# Admin contact for server issues
ServerAdmin webmaster@exampleMN.lab

# Define the web document root
DocumentRoot /var/www/html

# Log files and log level
ErrorLog logs/error_log
LogLevel info

# Load required modules
LoadModule mpm_prefork_module modules/mod_mpm_prefork.so
LoadModule unixd_module modules/mod_unixd.so
LoadModule systemd_module modules/mod_systemd.so
LoadModule log_config_module modules/mod_log_config.so
LoadModule mime_module modules/mod_mime.so
LoadModule authz_core_module modules/mod_authz_core.so

# Define user and group for the web service
User apache
Group apache

# Configure the default listening port
Listen 80
```

- Ensure the **apache** user and group exist before restarting the service.
- Run a syntax check on the configuration file:

  ```
  apachectl -t
  ```
- Restart Apache to apply changes:

  ```
  systemctl restart httpd
  ```

3. Testing the Apache Configuration

- Create a new default web page (index.html) inside /var/www/html/:

  ```
  <!DOCTYPE html>
  <html>
  <head>
      <title>NetworkID.exampleMN.lab</title>
  </head>
  <body>
      <h1>Host: hostname [IP_address:port]</h1>
  </body>
  </html>
  ```

- Test your Apache setup:
  - Open a browser and navigate to http://localhost.
  - Use wget to download the index page:

    ```
    wget http://localhost/index.html
    ```
  - Use **netcat** to test Apache manually:

    ```
    nc localhost 80
    GET /index.html HTTP/1.1
    Host: hostname
    ```

    (Press **Enter** twice to send the request.)
- Verify Apache accessibility from a client machine:

- Test hostname resolution using ping:

  ```
  ping server-hostname
  ```

- Use wget to fetch the page from another system:

  ```
  wget http://server-hostname/index.html
  ```

This ensures Apache is correctly installed, configured, and accessible across the network.

Section B – Configuring Directory Indexing

1. Enabling Directory Indexing

- Add the **DirectoryIndex** directive to your Apache configuration. This specifies the default file to be served when no filename is provided in the URL.
- Set index.html as the default resource.

Check the Apache documentation to confirm if a specific module is required for this directive. By default, index.html is used when the module is enabled and the directive is not explicitly set.

2. Testing Directory Indexing

- Open a web browser and enter your server's address **without specifying a file name**.
- Alternatively, use **netcat** to send a request:

  ```
  nc localhost 80
  ```

  ```
  GET / HTTP/1.0
  ```

(Press **Enter** twice to send the request.)

3. Enabling Optional Directory Listing

- If no `index.html` file exists, Apache can return a **directory listing** instead.
- To enable this feature, ensure the `mod_autoindex` module is loaded and add the following directive:

  ```
  Options Indexes
  ```

- This allows Apache to display a list of files within the directory when no default index file is present.

Section C – Configuring Virtual Websites

To host multiple websites using a single Apache instance, we will configure **virtual hosts**, allowing different sites to be served based on hostname or IP address. This is known as **virtual hosting**.

1. Understanding Virtual Hosting

- Apache supports multiple virtual hosting methods:
    - **Port-based**: Different sites run on different ports.
    - **IP-based**: Each site is assigned a unique IP address.
    - **Name-based**: Multiple sites share the same IP address but are distinguished by hostname.
- In this setup, we will use **name-based virtual hosting** (same IP, different hostnames) and also configure **one IP-based virtual host** on an aliased network interface.
- Virtual host definitions can be placed:
    - **Within the main Apache configuration file**
    - **In a separate configuration file** (referenced using Include <filename> or IncludeOptional).

Since virtual hosts rely on hostname resolution, **DNS configuration is required**.

2. Configuring DNS

- Configure your DNS server to resolve the following hostnames:
    - `www.exampleMN.lab`
    - `www.siteMN.lab`
    - `secure.exampleMN.lab`

    Question: How many forward zones need to be created or modified?
- Verify name resolution using forward and reverse DNS lookups.

3. Setting Up Name-Based Virtual Hosting

1. **Create a Directory Structure**
    - For each virtual site, create the following directory structure:

```
Mkdir -p /var/www/vhosts/www.exampleMN.lab/html /var/www/vhosts/www.exampleMN.lab/logs

Mkdir -p /var/www/vhosts/www.siteMN.lab/html /var/www/vhosts/www.siteMN.lab/logs
```

2. **Create Home Pages**
    - Inside each html directory, create an index.html file with:
        - Service name
        - Server hostname
        - IP address and port
        - Website name
3. **Modify Apache Configuration**
    - Add the following virtual host entries in the Apache configuration file:

```
<VirtualHost *:80>
    ServerName www.exampleMN.lab
    DocumentRoot /var/www/vhosts/www.exampleMN.lab/html
    ErrorLog /var/www/vhosts/www.exampleMN.lab/logs/error.log
    CustomLog /var/www/vhosts/www.exampleMN.lab/logs/access.log combined
</VirtualHost>


<VirtualHost *:80>
    ServerName www.siteMN.lab
    DocumentRoot /var/www/vhosts/www.siteMN.lab/html
    ErrorLog /var/www/vhosts/www.siteMN.lab/logs/error.log
    CustomLog /var/www/vhosts/www.siteMN.lab/logs/access.log combined
</VirtualHost>
```

    - Two key directives are required inside each VirtualHost block: *Can you identify them?*
    - (Optional) You may define separate error logs for each virtual host.
4. **Start Apache with the New Configuration**
    - Before restarting, validate the configuration:

```
apachectl -t
```

- Restart the Apache service:

  ```
  systemctl restart httpd
  ```

- Ensure Apache is listening on the correct ports.

5. **Test the Setup**
   - Access each site using a browser or command line:

     ```
     wget http://www.exampleMN.lab
     ```

     ```
     wget http://www.siteMN.lab
     ```

   - Refresh the page or clear the browser cache if needed.

4. Configuring IP-Based Virtual Hosting

1. **Create a Directory Structure**
   - For `secure.exampleMN.lab`:

     ```
     mkdir -p /var/www/vhosts/secure.exampleMN.lab/html
     /var/www/vhosts/secure.exampleMN.lab/logs
     ```

2. **Create a Unique Home Page**
   - Inside html, create index.html that includes:
     - Service name
     - Server hostname
     - IP address and port
     - Website name
3. **Modify Apache Configuration for IP-Based Virtual Host**
   - Assign an aliased interface for secure.exampleMN.lab:

```
<VirtualHost 192.168.1.100:80>
    ServerName secure.exampleMN.lab
    DocumentRoot /var/www/vhosts/secure.exampleMN.lab/html
    ErrorLog /var/www/vhosts/secure.exampleMN.lab/logs/error.log
    CustomLog /var/www/vhosts/secure.exampleMN.lab/logs/access.log combined
</VirtualHost>
```

   - Ensure all required directives are included.

4. **Start Apache and Verify the Setup**
    - Restart the Apache service and confirm it is listening on **all interfaces and port 80**.
    - Test all sites from both the server and a client:
        - **Localhost**: Should display the default page.
        - **Static IP**: Should display the first virtual host.
        - **Server hostname**:
            - From server → Default page
            - From client → First virtual host
        - **Each site name**: Should display its respective page.

By completing this configuration, you will have successfully set up **name-based and IP-based virtual hosts** on Apache, ensuring multiple websites can be hosted efficiently using automation techniques.

Section D – Configuring HTTPS

In this section, we will enable HTTPS for **secure.exampleMN.lab** on the aliased interface. The process includes:

- **Generating a Certificate**
- **Configuring SSL/TLS for Apache** to support an IP-based virtual host for **secure.exampleMN.lab**
    - Modifying the main Apache configuration to enable SSL/TLS
    - Creating an IP-based virtual host container on the aliased interface that supports secure connections

Background Information

SSL/TLS Overview

SSL/TLS ensures secure communication using multiple cryptographic techniques:

- **Asymmetric encryption** for key exchange
- **Symmetric encryption** for data confidentiality
- **Message authentication codes (MACs)** for data integrity

Key Distribution

- Symmetric encryption relies on a **shared secret**, often distributed via public-key cryptography.
- Public-key cryptography (asymmetric encryption) uses **certificates**, issued by **Certificate Authorities (CAs)**.
- The **Public Key Infrastructure (PKI)** governs certificate issuance and validation.

How PKI Works:

1. An organization generates a key pair (private & public key).
2. The public key, along with organization details, is submitted to a CA (e.g., Symantec, Entrust, GoDaddy).
3. The CA issues and digitally signs a certificate.
4. The organization makes this certificate publicly available.
5. Clients verify the certificate's authenticity using the CA's signature.

Types of Certificates:

- **Self-signed:** Used for testing; lacks third-party validation.
- **Local CA-signed:** Suitable for internal networks (Intranet).
- **Public CA-signed:** Trusted by external clients and browsers.

Creating a Self-Signed Certificate

We will generate a self-signed certificate using **OpenSSL**.

OpenSSL Command Options:

Option	Description
req	Request a certificate
-x509	Generate a self-signed certificate
-newkey rsa	Create a private RSA key
-days <n>	Set certificate validity period
-nodes	Generate a key **without** passphrase
-keyout <file>	Specify private key filename
-out <file>	Specify certificate filename

Steps to Generate the Certificate:

1. **Create the directory structure**:
2. `mkdir -p /etc/httpd/tls/{key,cert}`
3. `chmod 700 /etc/httpd/tls/key`
4. `chmod 755 /etc/httpd/tls/cert`
5. **Generate the private key and certificate:**
6. ```
   openssl req -x509 -newkey rsa -days 120 -nodes \
   -keyout /etc/httpd/tls/key/exampleMN.key \
   -out /etc/httpd/tls/cert/exampleMN.cert
   ```
7. **Enter the required details:**
   ```

- **Organization Name:** `CST8246`
- **Organizational Unit:** `exampleMN.lab`
- **Common Name:** `secure.exampleMN.lab` *(Must match the website URL for browsers to accept it)*

8. **Set file permissions:**
9. `chmod 600 /etc/httpd/tls/key/exampleMN.key`
10. `chmod 644 /etc/httpd/tls/cert/exampleMN.cert`
11. `chown root:root /etc/httpd/tls/{key,cert}/*`

Verifying the Certificate

To check certificate details:

```
openssl x509 -in /etc/httpd/tls/cert/exampleMN.cert -noout -text
```

To view the private key:

```
cat /etc/httpd/tls/key/exampleMN.key
```

To ensure the private key matches the certificate, compare their modulus values:

```
openssl x509 -noout -modulus -in /etc/httpd/tls/cert/exampleMN.cert | md5sum
openssl rsa -noout -modulus -in /etc/httpd/tls/key/exampleMN.key | md5sum
```

If both outputs are identical, the key and certificate match.

Configuring Apache for SSL/TLS

1. **Enable HTTPS in Apache by adding the following directives:**
   - **Ensure Apache listens on port 443:**

     ```
     Listen 443
     ```

   - **Load the SSL module (if not already enabled):**

     ```
     LoadModule ssl_module modules/mod_ssl.so
     ```

   - **Specify supported protocols:**

     ```
     SSLProtocol -all +TLSv1 +TLSv1.1 +TLSv1.2
     ```

2. **Define the SSL-enabled Virtual Host:**

```
<VirtualHost 192.168.1.100:443>
```

```
    ServerName secure.exampleMN.lab
    DocumentRoot "/var/www/vhosts/secure.exampleMN.lab/html"

    SSLEngine on
    SSLCertificateFile "/etc/httpd/tls/cert/exampleMN.cert"
    SSLCertificateKeyFile "/etc/httpd/tls/key/exampleMN.key"

    ErrorLog "/var/www/vhosts/secure.exampleMN.lab/logs/error.log"
    CustomLog "/var/www/vhosts/secure.exampleMN.lab/logs/access.log" combined
</VirtualHost>
```

3. **Run a syntax check & restart Apache:**
4. `apachectl configtest`
5. `systemctl restart httpd`

Testing the HTTPS Setup

1. **Verify Apache is listening on port 443:**

   ```
   netstat -tulnp | grep httpd
   ```

2. **Test HTTPS access using a web browser or CLI tools:**
   - From a browser:
   - https://secure.exampleMN.lab
   - Using wget or curl:

   ```
   wget --no-check-certificate https://secure.exampleMN.lab
   curl -k https://secure.exampleMN.lab
   ```

3. **Check Apache logs for errors:**

   ```
   tail -f /var/log/httpd/error_log
   ```

If everything is set up correctly, you should be able to access **secure.exampleMN.lab** securely over HTTPS.

Section E: Automating Apache Installation & Configuration

Note: *Automating Apache Installation & Configuration for Rocky and Debian is optional.*

Key Points:

- **Use automation scripts** to streamline Apache installation and virtual host setup.
- **Ensure compatibility** across RHEL (Rocky) and Debian systems.
- **Automate SSL/TLS** certificate generation and configuration.

Automation Exercise:

1. **Write a script** to:
    - Detect the OS type (RHEL/Rocky vs. Debian).
    - Install Apache using the appropriate package manager.
    - Configure virtual hosts automatically.
    - Enable and start the Apache service.
2. **Write a second script** to:
    - Install and configure SSL/TLS using mod_ssl (RHEL/Rocky) or openssl (Debian).
    - Generate a self-signed certificate.
    - Configure Apache to use HTTPS.
3. **Automate verification**:
    - Ensure Apache is running (`systemctl status httpd/apache2`).
    - Check listening ports (`netstat -tulnp or ss -tulnp`).
    - Verify logs for errors (`journalctl -xe and /var/log/httpd/error_log`).

Final Steps and Additional Notes

Double-Check Your Work

- Carefully review each step to ensure everything is set up correctly.
- If any issues arise, verify that no steps were skipped.

Refer to Official Documentation

- Comprehensive Apache HTTP Server documentation is available at: httpd.apache.org

Clear Browser Cache

- After modifying web repository data, clear your browser cache to reflect the latest changes.
- Refer to **Blackboard (BB)** for detailed cache-clearing instructions.

By following these steps, you'll ensure a smooth and secure HTTPS configuration for your web server.