Web Manual Lab 1 Report

Lab 1 – Apache Web Server Installation and Maintenance

---

1. Objective

The objective of this laboratory experiment is to **install, configure, and maintain the Apache Web Server** on a Linux-based system. This includes learning how to create and manage multiple virtual hosts, host static and dynamic web pages, and understand the working mechanism of the HTTP web service.

By the end of this lab, students should be able to:

- Install and verify the Apache server.
- Configure multiple websites using virtual hosts.
- Deploy and test simple dynamic web pages using HTML and JavaScript.
- Understand how local DNS mapping (/etc/hosts) affects virtual host access.

---

2. Software and Hardware Requirements

Hardware Requirements:

- A personal computer with at least 4 GB RAM and 20 GB free disk space
- Internet connection

Software Requirements:

- Operating System: Ubuntu / Debian / Kali Linux or any Debian-based distribution
- Web Server: Apache2
- Web Browser: Google Chrome / Mozilla Firefox
- Text Editor: VS Code / Nano / Vim

---

3. Theoretical Background

The **Apache HTTP Server** (commonly known as Apache) is one of the most widely used open-source web servers. It allows users to host websites, handle client requests, and deliver static or dynamic content over the HTTP or HTTPS protocols.

A **web server** listens on port 80 (HTTP) or 443 (HTTPS) and responds to requests from clients (browsers). Apache can serve multiple websites on the same machine using **Virtual Hosts** — each configured with its own domain name, document root, and log files.

Virtual Hosting allows multiple domains to be hosted on a single server. There are two main types:

- **Name-based virtual hosting:** Multiple domains share the same IP but are identified by their hostnames.
- **IP-based virtual hosting:** Each site has its own IP address.

---

4. Experimental Procedure

Task 1: Install Apache Web Server

1. **Update package list**
2. sudo apt update

This command fetches the latest package lists to ensure Apache installs with the latest version.

3. **Install Apache2**
4. sudo apt install -y apache2

This installs the Apache server and necessary dependencies.

5. **Enable and start the service**
6. sudo systemctl enable apache2
7. sudo systemctl start apache2

This ensures Apache automatically runs on boot.

8. **Check Apache status**
9. sudo systemctl status apache2

The output should show that Apache is active (running).

10. **Test installation**
    Open a browser and navigate to:
    http://localhost or http://127.0.0.1
    You should see the default “Apache2 Ubuntu Default Page”.

Checkpoint 1: The default Apache homepage loads successfully.

Task 2: Configure Virtual Hosts

1. **Create website directories**
2. sudo mkdir -p /var/www/example.com/html
3. sudo mkdir -p /var/www/webserverlab.com/html
4. sudo mkdir -p /var/www/anothervhost.com/html
5. **Assign ownership and permissions**
6. sudo chown -R $USER:$USER /var/www/
7. sudo chmod -R 755 /var/www/
8. **Create index pages for each site**

example.com/html/index.html

```
<html>
<body>
  <h1>Welcome to example.com</h1>
  <p>This is the first Apache virtual host.</p>
</body>
</html>
```

webserverlab.com/html/index.html

```
<html>
<body>
  <h1>Welcome to webserverlab.com</h1>
  <p>This is another virtual host.</p>
```

```
</body>
</html>
```

anothervhost.com/html/index.html

```
<html>
<body>
  <h1>Welcome to anothervhost.com</h1>
  <p>This is the third virtual host site.</p>
</body>
</html>
```

9. **Create configuration files**
   Create files in /etc/apache2/sites-available/ for each site:
10. sudo nano /etc/apache2/sites-available/example.com.conf

Example content:

```
<VirtualHost *:80>
    ServerAdmin admin@example.com
    ServerName example.com
    DocumentRoot /var/www/example.com/html
    ErrorLog ${APACHE_LOG_DIR}/example.com-error.log
    CustomLog ${APACHE_LOG_DIR}/example.com-access.log combined
</VirtualHost>
```

11. **Enable sites and reload Apache**
12. sudo a2ensite example.com.conf
13. sudo a2ensite webserverlab.com.conf
14. sudo a2ensite anothervhost.com.conf
15. sudo systemctl reload apache2
16. **Edit local hosts file**

17. sudo nano /etc/hosts

Add:

127.0.0.1 example.com webserverlab.com anothervhost.com

18. **Test in browser:**
    Open the following links:
    - http://example.com
    - http://webserverlab.com
    - http://anothervhost.com

Checkpoint 2: Each domain opens a different website.

Task 3: Deploy Dynamic Web Pages

1. **Modify index pages to include JavaScript:**
2. <!DOCTYPE html>
3. <html>
4. <head>
5. <title>Example.com</title>
6. <script>
7. function greet() {
8. const name = prompt("Enter your name:");
9. document.getElementById("msg").innerText = "Hello, " + name + "!";
10. }
11. </script>
12. </head>
13. <body>
14. <h1>Dynamic Site: example.com</h1>
15.

16. <button onclick="greet()">Click Me</button>
17. </body>
18. </html>
19. **Reload Apache server**
20. sudo systemctl restart apache2
21. **Open each domain in a browser** and interact with the page.

Checkpoint 3: Both dynamic sites function correctly.

---

5. Observations

Step	Action Performed	Output / Result
1	Installed Apache2	Apache service active
2	Created multiple virtual hosts	Each domain served unique content
3	Configured /etc/hosts	Domains resolved locally
4	Added JavaScript functionality	Dynamic behavior verified

---

6. Result

The Apache Web Server was successfully installed and configured.
Three virtual hosts (example.com, webserverlab.com, anothervhost.com) were created and tested.
Two dynamic web pages were deployed and verified for interactive functionality using JavaScript.

---

7. Conclusion

This lab demonstrated how to set up and manage a web server using Apache. Students learned about:

- The structure of Apache configuration files.
- Virtual hosting concepts.

- Local domain mapping.
- Deploying static and dynamic websites.

It provided hands-on understanding of real-world web hosting and development environments.