Shaeakh Ahmed Chowdhury

Reg. NO : 2020831022

Lab Report: Apache Web Server Installation & Maintenance

1. Objective

The goal of this lab was to install, configure, and maintain an **Apache Web Server** on Ubuntu and to understand its internal configuration and management.
Additionally, the task included **setting up virtual hosts** to serve multiple websites from a single server and **hosting simple dynamic websites** using **HTML and JavaScript**.

---

2. Approach

I followed a step-by-step method:

1. Installed the Apache server using Ubuntu's package manager.
2. Configured the firewall and verified that Apache was running successfully.
3. Created and managed **virtual hosts** (example.com, webserverlab.com, and anothervhost.com) to host multiple websites.
4. Learned key Apache management commands for starting, stopping, and reloading the server.
5. Hosted **two dynamic websites** with JavaScript handling user input on separate virtual hosts.
6. Verified functionality using browsers and local domain mapping.

---

3. Task 1: Installing Apache Web Server

Step 1: Installing Apache

First, I updated the system and installed Apache:

sudo apt update
sudo apt install apache2

Step 2: Adjusting the Firewall

I configured UFW to allow Apache traffic:

sudo ufw app list
sudo ufw allow 'Apache'
sudo ufw status

Output showed that **HTTP traffic (port 80)** was successfully allowed.

Step 3: Checking the Web Server

To confirm Apache was active:

sudo systemctl status apache2

It showed the service was **active (running)**.
Then, I opened the default page in a browser via:

http://127.0.0.1
http://localhost
http://webserverlab.com

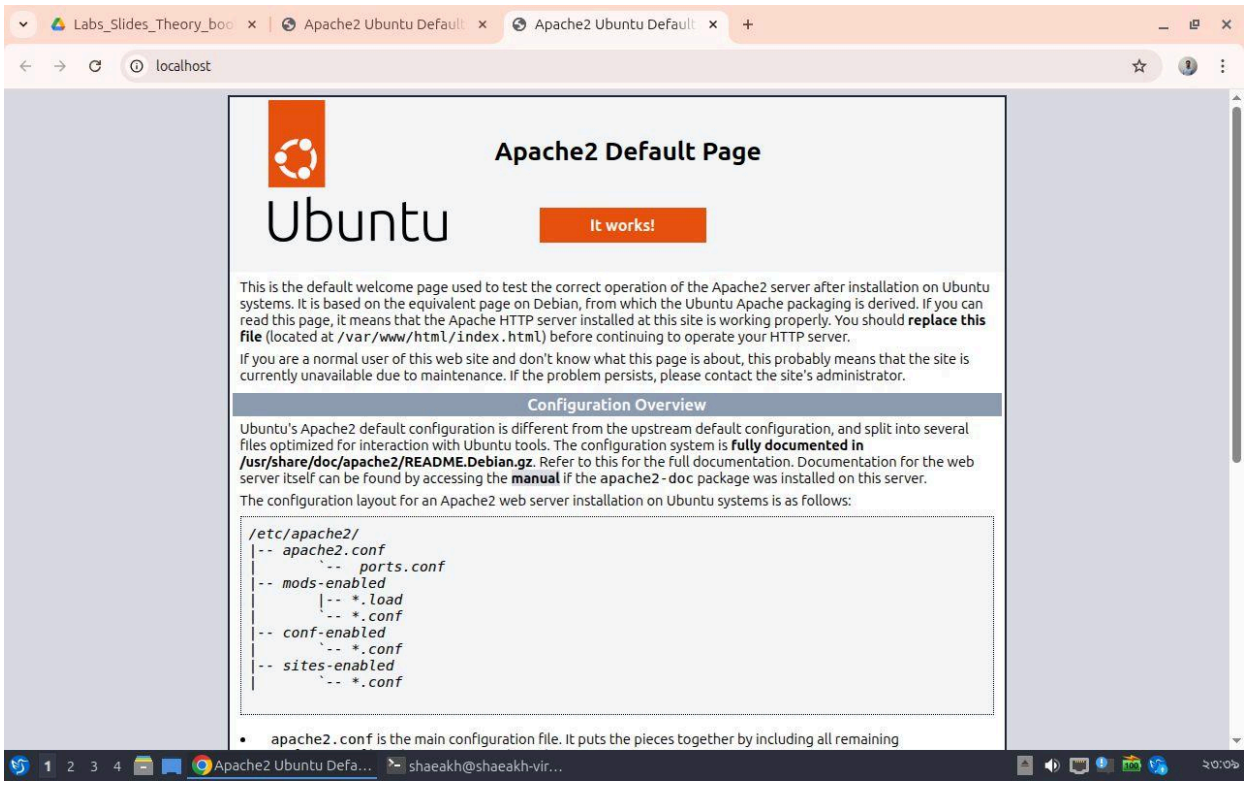
For local domain mapping:

sudo nano /etc/hosts

Added:

127.0.0.1 webserverlab.com

Checkpoint 1 completed: Apache installed and verified successfully.



---

4. Task 2: Setting Up Virtual Hosts

Step 1: Managing Apache Process

I learned to control Apache using the following commands:

sudo systemctl stop apache2
sudo systemctl start apache2
sudo systemctl restart apache2
sudo systemctl reload apache2
sudo systemctl disable apache2
sudo systemctl enable apache2

Step 2: Creating a Virtual Host (example.com)

Created a directory for the site:

```
sudo mkdir -p /var/www/example.com/html
sudo chown -R $USER:$USER /var/www/example.com/html
sudo chmod -R 755 /var/www/example.com
```

Created a simple web page:

```
nano /var/www/example.com/html/index.html
```

HTML Code:

```
<html>
<head>
<title>Welcome to Example.com!</title>
</head>
<body>
<h1>Success! The example.com server block is working!</h1>
</body>
</html>
```

Created a configuration file:

```
sudo nano /etc/apache2/sites-available/example.com.conf
```

Configuration:

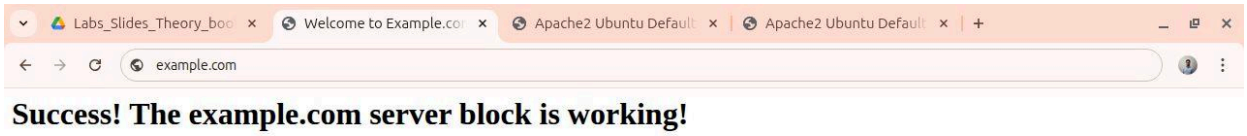
```
<VirtualHost *:80>
    ServerAdmin admin@example.com
    ServerName example.com
    ServerAlias www.example.com
    DocumentRoot /var/www/example.com/html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

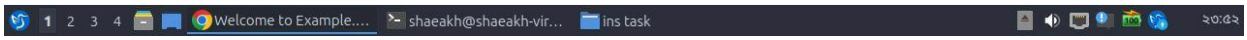
Enabled the site and disabled the default one:

```
sudo a2ensite example.com.conf
sudo a2dissite 000-default.conf
sudo apache2ctl configtest
```

sudo systemctl restart apache2

Visiting **http://example.com** displayed my custom page.
Checkpoint 2 completed.





Step 3: Testing Multiple Virtual Hosts

To add another virtual host:

sudo mkdir -p /var/www/anothervhost.com/html
sudo chown -R $USER:$USER /var/www/anothervhost.com/html

Created HTML:

nano /var/www/anothervhost.com/html/index.html

Code:

<html>

```
<head>
<title>Welcome to AnotherVHost!</title>
</head>
<body>
<h1>This is Another Virtual Host running on Apache!</h1>
</body>
</html>
```

Configured virtual host:

sudo nano /etc/apache2/sites-available/anothervhost.com.conf

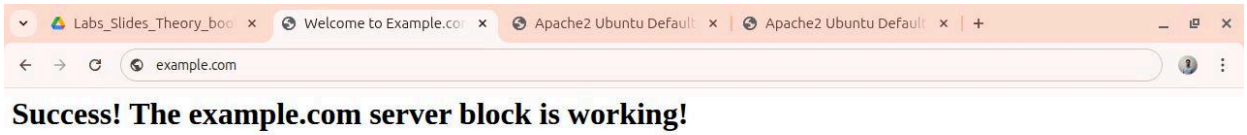
Configuration:

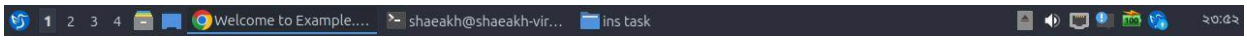
```
<VirtualHost *:80>
    ServerAdmin admin@anothervhost.com
    ServerName anothervhost.com
    DocumentRoot /var/www/anothervhost.com/html
</VirtualHost>
```

Enabled and restarted:

sudo a2ensite anothervhost.com.conf
sudo systemctl restart apache2

Checkpoint 4 completed: Multiple virtual hosts successfully configured.





---

5. Task 3: Hosting Dynamic Websites using HTML & JavaScript

For this part, I hosted **two simple dynamic websites**.
Each was placed under a different virtual host directory.

Website 1: Calculator

Path: `/var/www/example.com/html/index.html`

```
<html>
<head>
<title>Simple Calculator</title>
<script>
function calculate() {
  const num1 = parseFloat(document.getElementById('num1').value);
  const num2 = parseFloat(document.getElementById('num2').value);
  const result = num1 + num2;
```

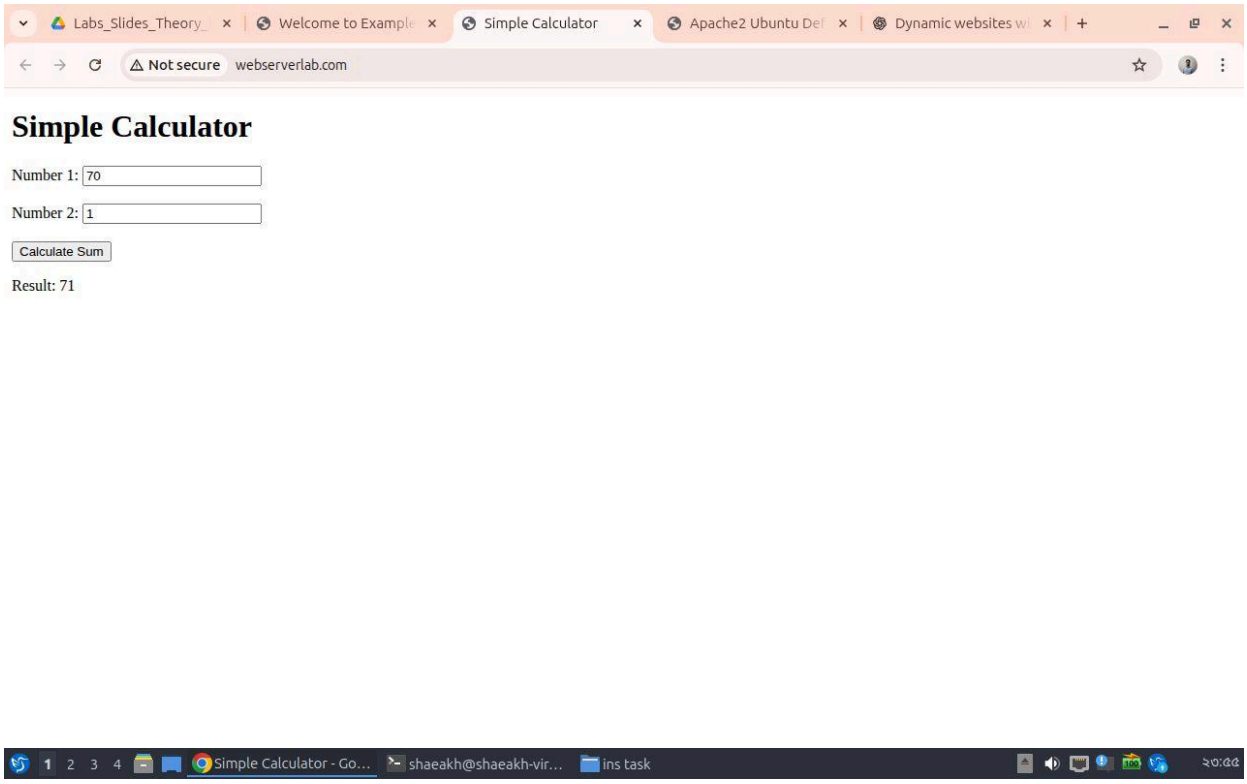
```
  document.getElementById('output').innerText = "Result: " + result;
}
</script>
</head>
<body>
<h2>Simple Calculator</h2>
<input id="num1" type="number" placeholder="Enter number 1">
<input id="num2" type="number" placeholder="Enter number 2">
<button onclick="calculate()">Add</button>
<p id="output"></p>
</body>
</html>
```

Website 2: Temperature Converter

Path: `/var/www/anothervhost.com/html/index.html`

```
<html>
<head>
<title>Temperature Converter</title>
<script>
function convert() {
  const celsius = document.getElementById('celsius').value;
  const fahrenheit = (celsius * 9/5) + 32;
  document.getElementById('result').innerText = `${fahrenheit} °F`;
}
</script>
</head>
<body>
<h2>Celsius to Fahrenheit Converter</h2>
<input id="celsius" type="number" placeholder="Enter Celsius">
<button onclick="convert()">Convert</button>
<p id="result"></p>
</body>
</html>
```

Checkpoint 5 completed: Two dynamic sites successfully hosted and tested.



7. Observations

- Apache service management is simple once systemctl commands are understood.
- Virtual hosts allow efficient hosting of multiple domains on a single server.
- Modifying `/etc/hosts` is helpful for local testing without a DNS.
- JavaScript can add interactivity even without a backend.
- Proper file ownership and permissions (`chown`, `chmod`) are crucial to avoid “403 Forbidden” errors.

---

8. Conclusion

This lab gave me practical exposure to how web servers work under the hood.
I successfully:

- Installed and configured Apache on Ubuntu.
- Created and managed multiple virtual hosts.
- Deployed two dynamic websites using only HTML and JavaScript.

The lab deepened my understanding of **web server architecture**, **configuration files**, and **domain hosting principles**.

---

9. References (Sources)

1. DigitalOcean Tutorial: How To Install Apache on Ubuntu 18.04
2. JavaWorld: Using JavaScript and Forms
3. Apache HTTP Server Documentation – https://httpd.apache.org/docs/
4. Ubuntu Man Pages – https://manpages.ubuntu.com/