Lab 7: Linux Web Server Lab: Apache & Virtual Hosting

Web servers are a fundamental building block of modern Internet applications. In this lab, you will learn how to install and configure the Apache web server on Ubuntu 24.04 LTS. Apache is one of the most widely used open-source HTTP servers in the world and is a common component in many web application stacks (LAMP, for example).

The goal of this exercise is to give you **hands-on experience** with:

- Installing Apache on a Linux server
- Understanding the default web root and serving a simple HTML page
- Configuring Apache virtual hosts to run multiple websites on the same server
- Mapping domain names (e.g. gci.example.com) to your server using the /etc/hosts file

By completing this lab, you will gain practical skills in basic web server administration, configuration management, and name resolution, which are essential for backend development, DevOps, and cloud/IoT deployments.

Objective

Install, configure, and deploy a functional Apache web server on Ubuntu 24 LTS, then connect it to a verification endpoint.

Task Instructions

Part 1: System Preparation

1.1 Update your system package list:

```
sudo apt update
```

1.2 Upgrade existing packages:

```
sudo apt upgrade -y
```

Part 2: Apache Installation

2.1 Install Apache2 web server:

```
sudo apt install apache2 -y
```

2.2 Verify Apache service status:

```
sudo systemctl status apache2
```

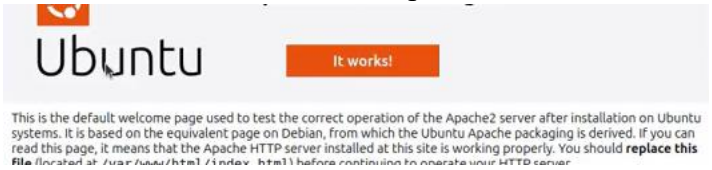
2.3 Enable Apache to start on boot:

```
sudo systemctl enable apache2
```

2.4 Check Enable Apache :

```
Enabling site 000-default.
```

2.5 Confirm Apache is working by opening a new page in your browser and navigating to http://localhost. You should see the "Apache2 Ubuntu Default Page."



Checkpoint :
Screenshot of the Apache default page in a browser.

Part 3: Serve Your Own HTML Page from the Default Site

By default, Apache comes with a basic site (the one that we saw in the previous step) enabled. We can modify its content in /var/www/html or settings by editing its Virtual Host file found in /etc/apache2/sites-enabled/000-default.conf.

3.1 Go to the web root:

```
cd /var/www/html
```

3.2 Create your own HTML file (e.g. my-page.html):

```
sudo nano my-page.html
```

3.3 Put **your own content** in the file, for example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>My First Apache Page</title>
```

```
</head>
<body>
    <h1>Apache Test Page</h1>
    <p>Student: [Your Name]</p>
    <p>Index number: [Your Index]</p>
    <p>Server: Ubuntu 24.04 LTS with Apache2</p>
</body>
</html>
```

3.4 Save and exit (`Ctrl+O`, `Enter`, then `Ctrl+X` in nano).
3.5 Access it via browser:

```
http://localhost/my-page.html
```

Checkpoint – Deliverable:
Screenshot of your custom `my-page.html` shown in the browser.

Part 4: Create and Configure a New Virtual Host

You will create a new Apache virtual host configuration based on the default one.

4.1 Go to the site configuration directory:

```
cd etc/apache2/sites-available
```

4.2 List the files:

You should see `000-default.conf`.

4.3 Copy the default configuration to a new file (e.g. `gci.conf`):

```
sudo cp 000-default.conf gci.conf
```

4.4 Edit the new config file:

```
sudo nano gci.conf
```

4.5 Modify it as follows:

- Set `ServerAdmin` to your email, e.g.:

```
ServerAdmin your.email@edu.p.lodz.com
```

- Add/modify `ServerName` to:

```
ServerName gci.example.com
```

- Change `DocumentRoot` to point to your new directory:

```
DocumentRoot /var/www/gci
```

- The relevant section should look roughly like:

```
<VirtualHost *:80>
    ServerAdmin your.email@domain.com
    ServerName gci.example.com

    DocumentRoot /var/www/gci

    ErrorLog ${APACHE_LOG_DIR}/gci-error.log
    CustomLog ${APACHE_LOG_DIR}/gci-access.log combined
</VirtualHost>
```

- Save and exit.
- Enable the new site:

```
sudo a2ensite gci.conf
```

- (Optional but recommended) Disable the default site:

```
sudo a2dissite 000-default.conf
```

- Test Apache configuration syntax:

```
sudo apache2ctl configtest
```

Expected: `Syntax OK`

- Reload Apache to apply changes:

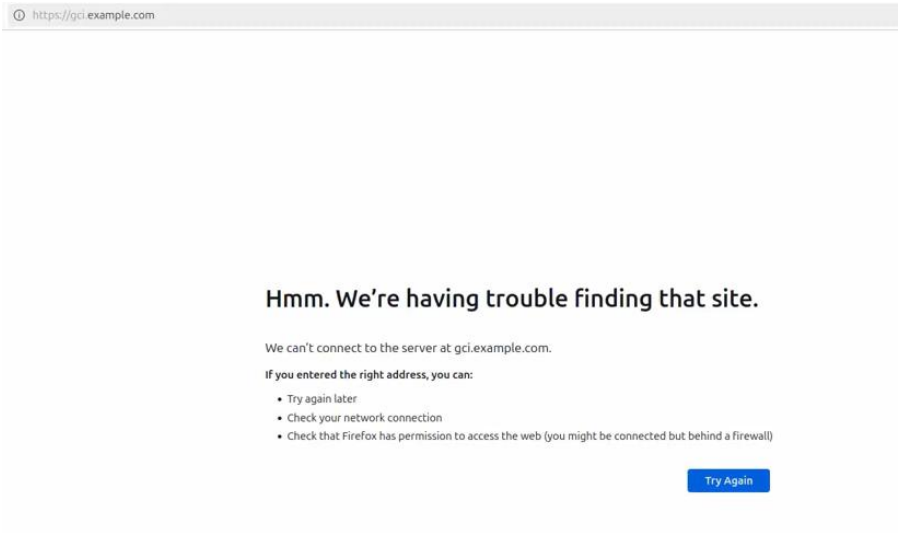
```
sudo systemctl reload apache2
```

Checkpoint – Deliverable:
Screenshot of gci.conf (or terminal showing cat /etc/apache2/sites-available/gci.conf) and output of apache2ctl configtest.

Part 5: Map Server Name to IP Using `/etc/hosts`

How to access `gci.example.com` from your browser without real DNS? you will use the host's file.

Challenge point for 5 grades



Final Questions (Short Answers)

Add to your lab report:

1. What is the purpose of a virtual host in Apache?
2. Why do we need to edit /etc/hosts in this lab, and when is DNS needed instead?
3. What is the difference between the default Apache site and the gci.example.com site you created?
4. What potential security risks exist if you misconfigure DocumentRoot or permissions in /var/www?