Lab # 01
Apache Server Setup and Locat Host Configuration

Apache Web Server is a software package that turns a computer into an HTTP server. That is, it sends web pages – stored as HTML files – to people on the internet who request them. It is open-source software, which means it can be used and modified freely.

How to Install Apache on Ubuntu

Open a terminal and type:

sudo apt-get update

Step 1: Install Apache

To install the Apache package on Ubuntu, use the command:

sudo apt-get install apache2

The system prompts for confirmation – do so, and allow the system to complete the installation.

```
sabahat@sabahat-Inspiron-7558:~$ sudo apt-get install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  javascript-common libjs-jquery libjs-sphinxdoc libjs-underscore
  libpython-all-dev libpython-dev libpython2-dev libpython2.7 libpython2.7-dev
  linux-headers-5.3.0-51 linux-headers-5.3.0-51-generic
  linux-image-5.3.0-51-generic linux-modules-5.3.0-51-generic
  linux-modules-extra-5.3.0-51-generic python-dbus python2-dev python2.7-dev
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap
0 upgraded, 8 newly installed, 0 to remove and 56 not upgraded.
Need to get 1,714 kB of archives.
After this operation, 7,493 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Step 2: Verify Apache Installation

To verify Apache was installed correctly, open a web browser and type in the address bar:

http://local.server.ip

or

localhost

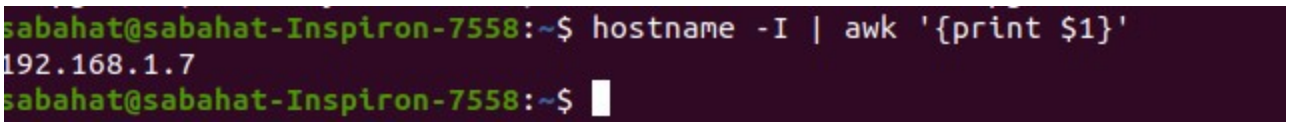
or 127.0.0.1

The web browser should open a page labeled “Apache2 Ubuntu Default Page,” as in the image below:

Replace local.server.ip with the IP address of your server. If you are unsure what's the IP address, run the following terminal command:

hostname -I | awk '{print $1}'

The output will return your server's IP address.



Or you can also check IP through command :
tfconfig

```
sabahat@sabahat-Inspiron-7558:~$ ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 6243  bytes 553412 (553.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6243  bytes 553412 (553.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp1s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.7  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::2cc1:706e:125b:43b6  prefixlen 64  scopeid 0x20<link>
        ether 94:65:9c:a2:76:9b  txqueuelen 1000  (Ethernet)
        RX packets 82446  bytes 108023186 (108.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 75071  bytes 9034177 (9.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

netstat

netstat (network statistics) is a **command** line tool for monitoring network connections both incoming and outgoing as well as viewing routing tables, interface statistics etc.

```
sabahat@sabahat-Inspiron-7558:~$ sudo systemctl start apache2.service
[sudo] password for sabahat:
sabahat@sabahat-Inspiron-7558:~$ netstat -ntulp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      -
tcp6       0      0 :::80                   :::*                    LISTEN      -
tcp6       0      0 :::1716                 :::*                    LISTEN      1967/kdeconnectd
tcp6       0      0 ::1:631                 :::*                    LISTEN      -
udp        0      0 127.0.0.53:53           0.0.0.0:*                           -
udp        0      0 192.168.1.7:68          0.0.0.0:*                           -
udp        0      0 0.0.0.0:631             0.0.0.0:*                           -
udp        0      0 0.0.0.0:5353            0.0.0.0:*                           -
udp        0      0 0.0.0.0:59251           0.0.0.0:*                           -
udp6       0      0 :::5353                 :::*                                -
udp6       0      0 :::1716                 :::*                                1967/kdeconnectd
udp6       0      0 :::39202                :::*                                -
sabahat@sabahat-Inspiron-7558:~$
```

Step 3: Configure Your Firewall

Although the Apache installation process is complete, there is one more additional step. Configure the default UFW (Uncomplicated Firewall, a software application that blocks network traffic (usually for security)) firewall to allow traffic on port 80.

Start by displaying available app profiles on UFW:

sudo ufw show app list

```
sabahat@sabahat-Inspiron-7558:~$ sudo ufw app list
Available applications:
  Apache
  Apache Full
  Apache Secure
  CUPS
```

Use the following command to allow normal web traffic on port 80:sudo ufw allow 'Apache'

```
sabahat@sabahat-Inspiron-7558:~$ sudo ufw allow 'Apache'
Rules updated
Rules updated (v6)
```

Verify the changes by checking UFW status:sudo ufw status

```
sabahat@sabahat-Inspiron-7558:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
Apache                     ALLOW       Anywhere
Apache (v6)                ALLOW       Anywhere (v6)
```

Apache Configuration

Apache Service Controls

When managing a web server, it’s helpful to have some level of control over the service. You’ll probably find yourself reloading or restarting Apache quite frequently, as you make configuration changes and test them. However, it’s also helpful to be able to stop (and start) the Apache service as needed.

This operation uses the systemctl command, with a series of switches:

Stop Apache:

sudo systemctl **stop** apache2.service

Start Apache:

sudo systemctl **start** apache2.service

Restart Apache:

sudo systemctl **restart** apache2.service

Reload Apache:

sudo systemctl **reload** apache2.service

Apache Configuration Files, Directories and Modules

Now that you have Apache installed, there are a couple of other things you will need to be aware of to make content available online. Most of all, this means dealing with directories and configuration files.

Directories

After installing, Apache by default creates a document root directory at /var/www/html**.**

Any files that you place into this directory are available to Apache to distribute over the network. Which means, this is the place where you copy web page files you want to publish. This is also where you would want to install content management systems, such as WordPress.

Configuration Files

As mentioned above, website content is stored in the/var/www/html/directory. You can create subdirectories within this location for each different website hosted on your server.

Apache creates log files for any errors it generates in the file /var/log/apache2/error.log.

It also creates access logs for its interactions with clients in the file /var/log/apache2/access.log.

Like many Linux-based applications, Apache functions through the use of configuration files. They are all located in the /etc/apache2/ directory.

Here's a list of other essential directories:

- /etc/apache2/apache2.conf – This is the main Apache configuration file and controls everything Apache does on your system. Changes here affect all the websites hosted on this machine.
- /etc/apache2/ports.conf – The port configuration file. You can customize the ports Apache monitors using this file. By default, **Port 80** is configured for http traffic.
- /etc/apache2/sites-available – Storage for Apache virtual host files. A virtual host is a record of one of the websites hosted on the server.
- /etc/apache2/sites-enabled – This directory holds websites that are ready to serve clients. The a2ensite command is used on a virtual host file in the **sites-available** directory to add sites to this location.

Modules

If you intend to work with software modules – applications that expand or enhance the functionality of Apache – you can enable them by using:

sudo a2enmod *name_of_module*

To disable the module:

sudo a2dismod *name_of_module*

What is Apache Virtual Host?

Virtual Host allows you to run multiple websites from a single physical server or Virtual Private Server. There are two types of virtual hosts on Apache:

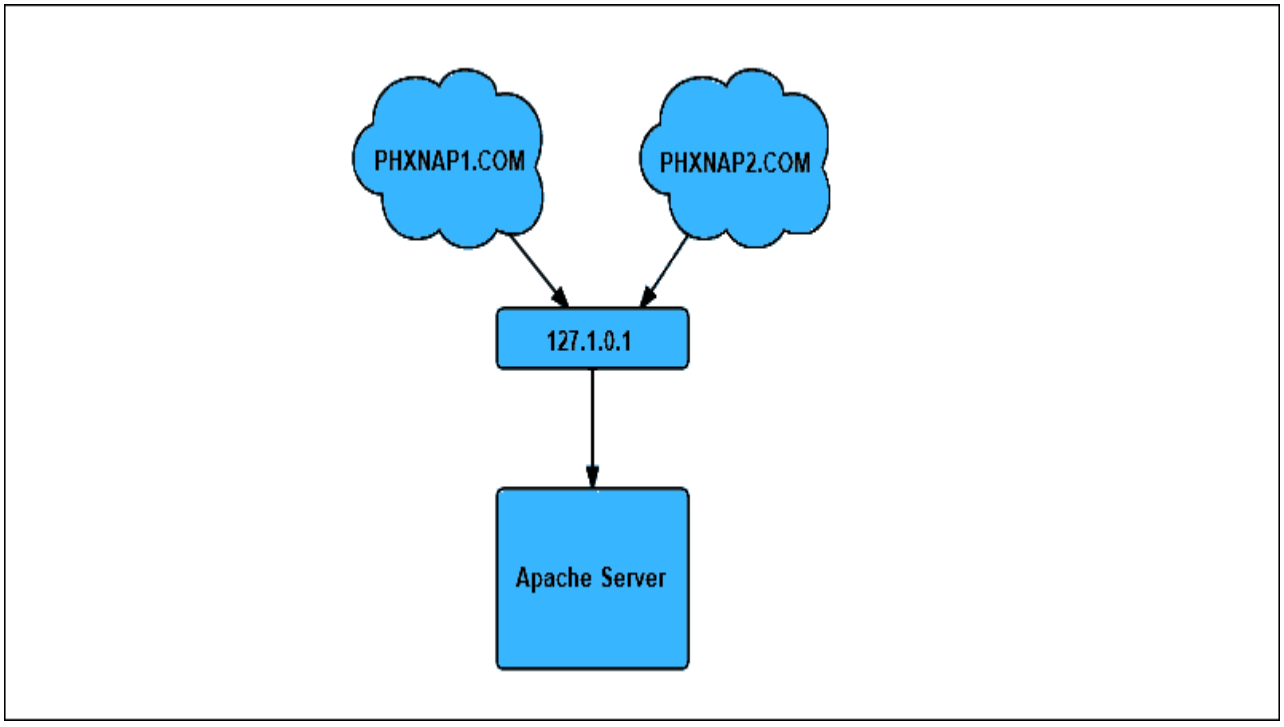
- **IP-Based Virtual Hosting** – every individual website on the Apache Server uses a different, unique IP address.
- **Name-Based Virtual Hosts** – enables you to add multiple domains using a single IP address.

Set Up Multiple Domains on a Single IP

Apache Virtual Host allows you to maximize your resources when setting up a website. With this powerful software, can use a single server and a single IP address to host a number of different domains.

How to Set Up a Name-Based Virtual Host

Name-based virtual hosting allows the client to report the hostname to the server, as an element of the HTTP header. This feature means that one machine can host multiple websites sharing the same IP address.



Step 1: Create a Directory Structure

Each virtual host needs to have a directory for storing virtual host data. Create directories and a directory structure at the following location */var/www*. In our example, we've created *phxnap1.com* and *phxnap2.com* directories, one for each domain name.

1. Enter the following command and replace the example domain with your domain names:

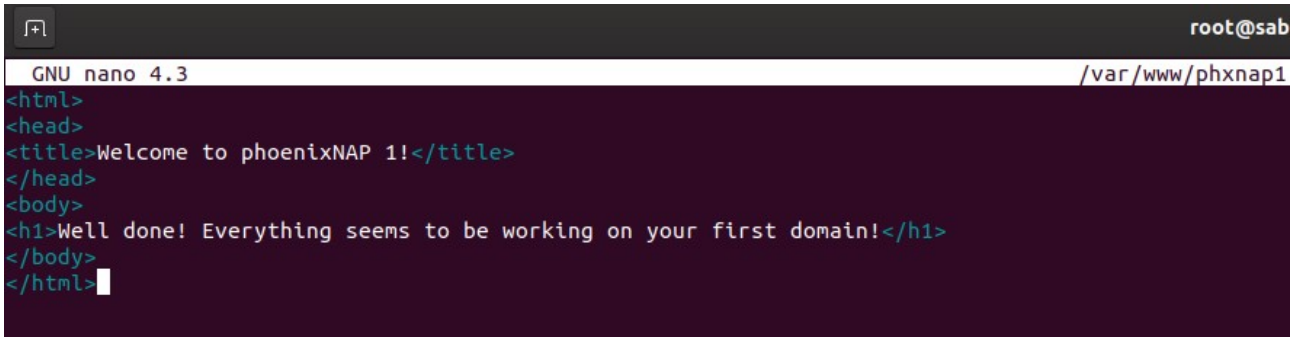
```
sudo mkdir -p /var/www/phxnap1.com/public_html

sudo mkdir -p /var/www/phxnap2.com/public_html
```

Within the directories, we also created *public_html*. These directories are going to store website files for the domains

2. Next, create a sample *index.html* page for each domain, using *nano* or your favorite text editor. Start with the first domain:

```
nano /var/www/phxnap1.com/public_html/index.html
```



4. Save and exit the file.5. Next, create a sample page for the second domain:

```
nano /var/www/phxnap2.com/public_html/index.html
```

6. Add the following lines to the file:

```
<html>
<head>
<title>Welcome to phoenixNAP 2!</title>
</head>
<body>
<h1>Well done! Everything seems to be working on your second domain!</h1>
</body>
</html>
```

7. Save and exit the second HTML file.

8. To prevent any permission issues modify the ownership of your document's root directory to *www-data*. The chown command is useful in this instance:

```
sudo chown –R www-data:www-data /var/www/phxnap1.com
```

```
sudo chown –R www-data:www-data /var/www/phxnap2.com
```

Step 2: Create a Virtual Host Configuration File

Apache Virtual Host configuration files are stored in the */etc/apache2/sites-available* directory.

1. To create a basic configuration file for your first domain, enter the domain information in the command:

```
sudo nano /etc/apache2/sites-available/phxnap1.com.conf
```

2. Add the following configuration block to create a basic configuration file. This example uses the first test domain, *phxnap1.com*. Make sure to enter the correct domain for your website:

```
<VirtualHost *:80>

ServerAdmin webmaster@phxnap1.com
ServerName phxnap1.com
ServerAlias www.phxnap1.com
DocumentRoot /var/www/phxnap1.com/public_html

ErrorLog ${APACHE_LOG_DIR}/phxnap1.com-error.log
CustomLog ${APACHE_LOG_DIR}/phxnap1.com-access.log combined

</VirtualHost>
```

- **ServerName** – represents the domain
- **ServerAlias** – represents all other domains such as subdomains
- **DocumentRoot** – the directory used by Apache to serve domain files
- **ErrorLog, CustomLog** – designates the log files location

There is no established format. However, naming your configuration files based on the domain name is a “best practice”.

4. Once you have edited the config file for the first domain, repeat the process for the rest. In our case, we will run:

```
sudo nano /etc/apache2/sites-available/phxnap2.com.conf
```

5. Then, add the configuration block as in the example above, making sure to change the values for *phxnap2.com*:

```
<VirtualHost *:80>

ServerAdmin webmaster@phxnap2.com
ServerName phxnap2.com
ServerAlias www.phxnap2.com
DocumentRoot /var/www/phxnap2.com/public_html

ErrorLog ${APACHE_LOG_DIR}/phxnap2.com-error.log
CustomLog ${APACHE_LOG_DIR}/phxnap2.com-access.log combined

</VirtualHost>
```

Step 3: Enable Virtual Host Configuration Files

To enable the virtual host file, create a symbolic link from the virtual host file to the sites-enabled directory. Apache2 reads this file when staring.

1. Use the *a2ensite helper* to enable the virtual host file with the command:

sudo a2ensite phxnap1.com

```
root@sabahat-Inspiron-7558:~# sudo a2ensite phxnap1.com
Enabling site phxnap1.com.
To activate the new configuration, you need to run:
  systemctl reload apache2
```

2. Repeat the process for the second domain by typing:sudo a2ensite phxnap2.com

```
Enabling site phxnap2.com.
To activate the new configuration, you need to run:
  systemctl reload apache2
root@sabahat-Inspiron-7558:~#
```

3. Next, verify the configuration file syntax is correct using the command:sudo apachectl configtest

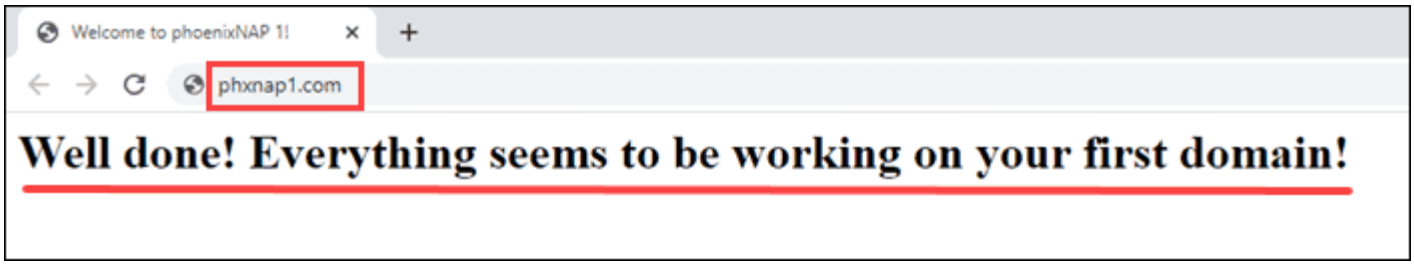
The message in the terminal will confirm that the syntax is correct: *"Syntax OK"*

4. Restart Apache2 for the changes to be applied:

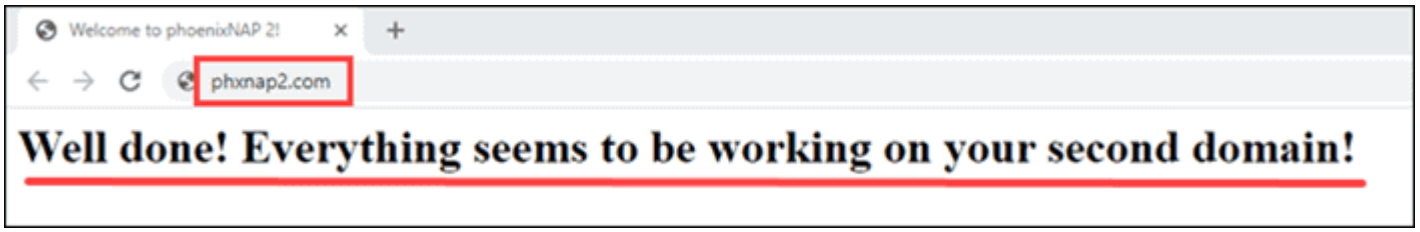
sudo systemctl restart apache2

5. The only thing left to do is to go to a web browser and access your websites. Based on the *index.html* file we created earlier, the appropriate message should appear for each domain.

phxnap1.com



phxnap2.com



Step 4: Configure Firewall (Optional)

Modify your firewall settings to improve security by creating a rule to enable SSH on Ubuntu:

sudo ufw allow OpenSSH

Add the rules to allow access to Apache.

sudo ufw allow in “Apache Full”

Next, enable the firewall.

sudo ufw enable