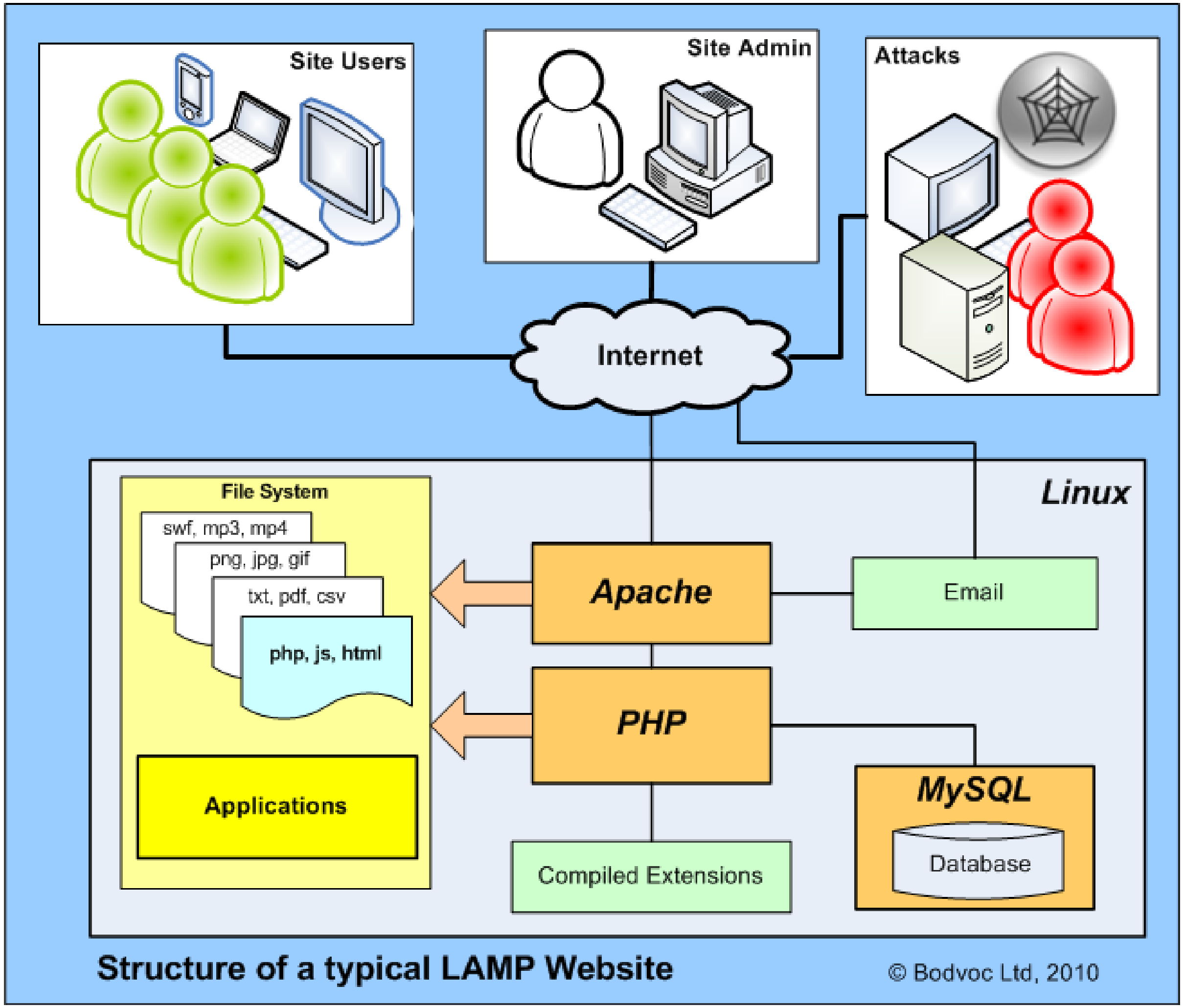
# Topic 09B
# HTTP

# Sample Web Application

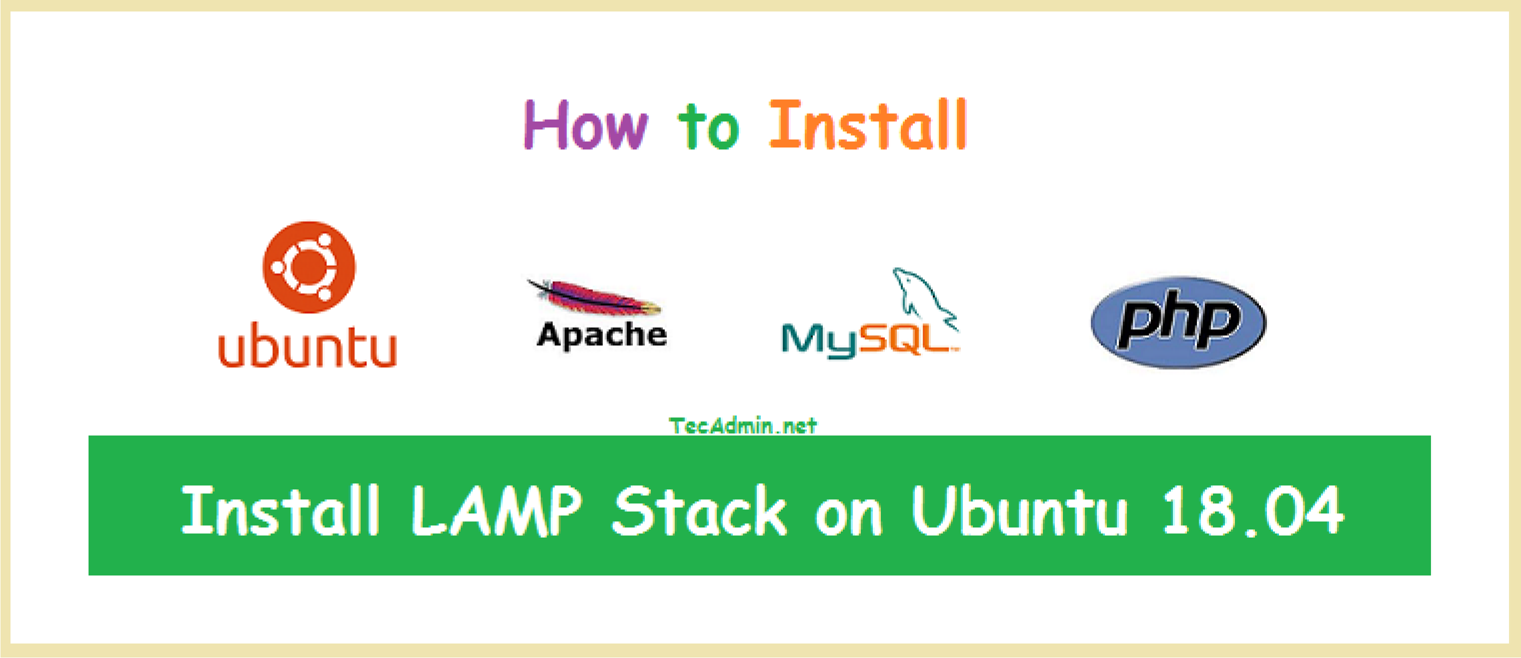


Structure of a typical LAMP Website

© Bodvoc Ltd, 2010

# What is LAMP?

https://tecadmin.net/install-lamp-stack-on-ubuntu-18-04/



# Web Server

## HTTPD - Apache2 Web Server

Apache is the most commonly used Web server on Linux systems. Web servers are used to serve Web pages requested by client computers. Clients typically request and view Web pages using Web browser applications such as *Firefox*, *Opera*, *Chromium*, or *Internet Explorer*.

Users enter a Uniform Resource Locator (URL) to point to a Web server by means of its Fully Qualified Domain Name (FQDN) and a path to the required resource. For example, to view the home page of the Ubuntu Web site a user will enter only the FQDN:

```
www.ubuntu.com
```

To view the community sub-page, a user will enter the FQDN followed by a path:

```
www.ubuntu.com/community
```

The most common protocol used to transfer Web pages is the Hyper Text Transfer Protocol (HTTP). Protocols such as Hyper Text Transfer Protocol over Secure Sockets Layer (HTTPS), and File Transfer Protocol (FTP), a protocol for uploading and downloading files, are also supported.

Apache Web Servers are often used in combination with the *MySQL* database engine, the HyperText Preprocessor (*PHP*) scripting language, and other popular scripting languages such as *Python* and *Perl*. This configuration is termed LAMP (Linux, Apache, MySQL and Perl/Python/PHP) and forms a powerful and robust platform for the development and deployment of Web-based applications.https://help.ubuntu.com/lts/serverguide/httpd.html

# Install Apache web server

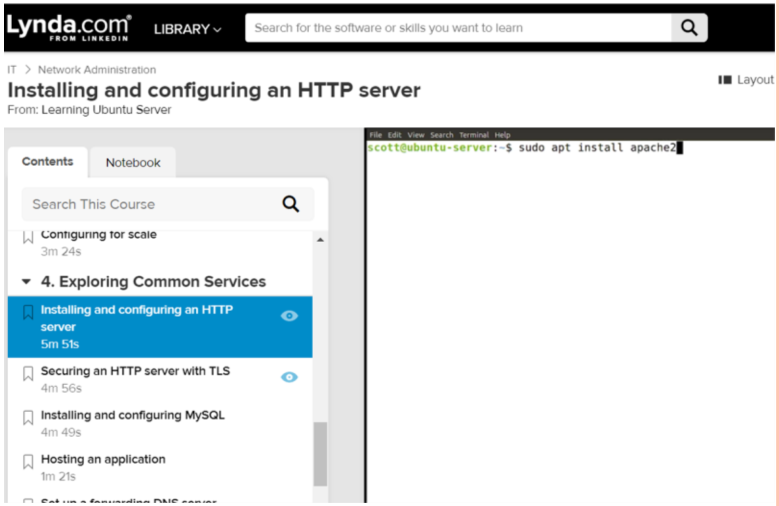
Install the HTTP server the command

```
sudo apt-get update

 sudo apt-get install apache2
```

Check to see if apache2 service is running:

```
        sudo systemctl status apache2
```


Lynda.com FROM LINKEDIN
LIBRARY
Search for the software or skills you want to learn
Layout
Contents
Notebook
Search This Course
File Edit View Search Terminal Help

IT > Network Administration

Installing and configuring an HTTP server

From: Learning Ubuntu Server

```
scott@ubuntu-server:~$ sudo apt install apache2
```

Configuring for scale
3m 24s

4. Exploring Common Services

Installing and configuring an HTTP server
5m 51s

Securing an HTTP server with TLS
4m 56s

Installing and configuring MySQL
4m 49s

Hosting an application
1m 21s

Set up a forwarding DNS server

# HTTPd

- Install **httpd**

- Put your desired content into /var/www/html

- restorecon –rv  /var/www/html to fix selinux context

- Edit /etc/httpd/conf/httpd.conf

# HTTPd

- Common options in config file

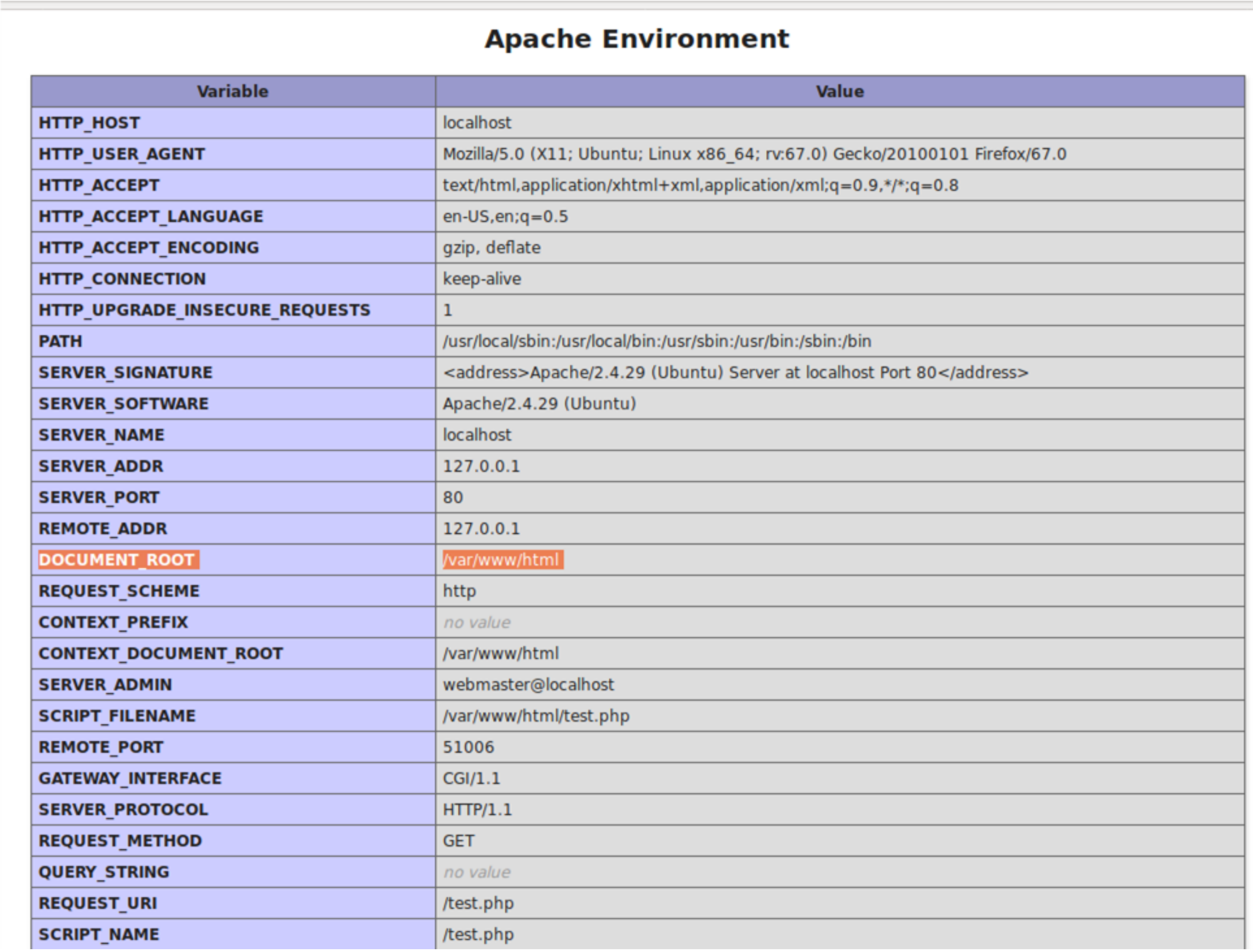
  DirectoryIndex – Determine file that is the index

  DocumentRoot – The base of the content folder

  <Directory /> – Directive determined by the directory

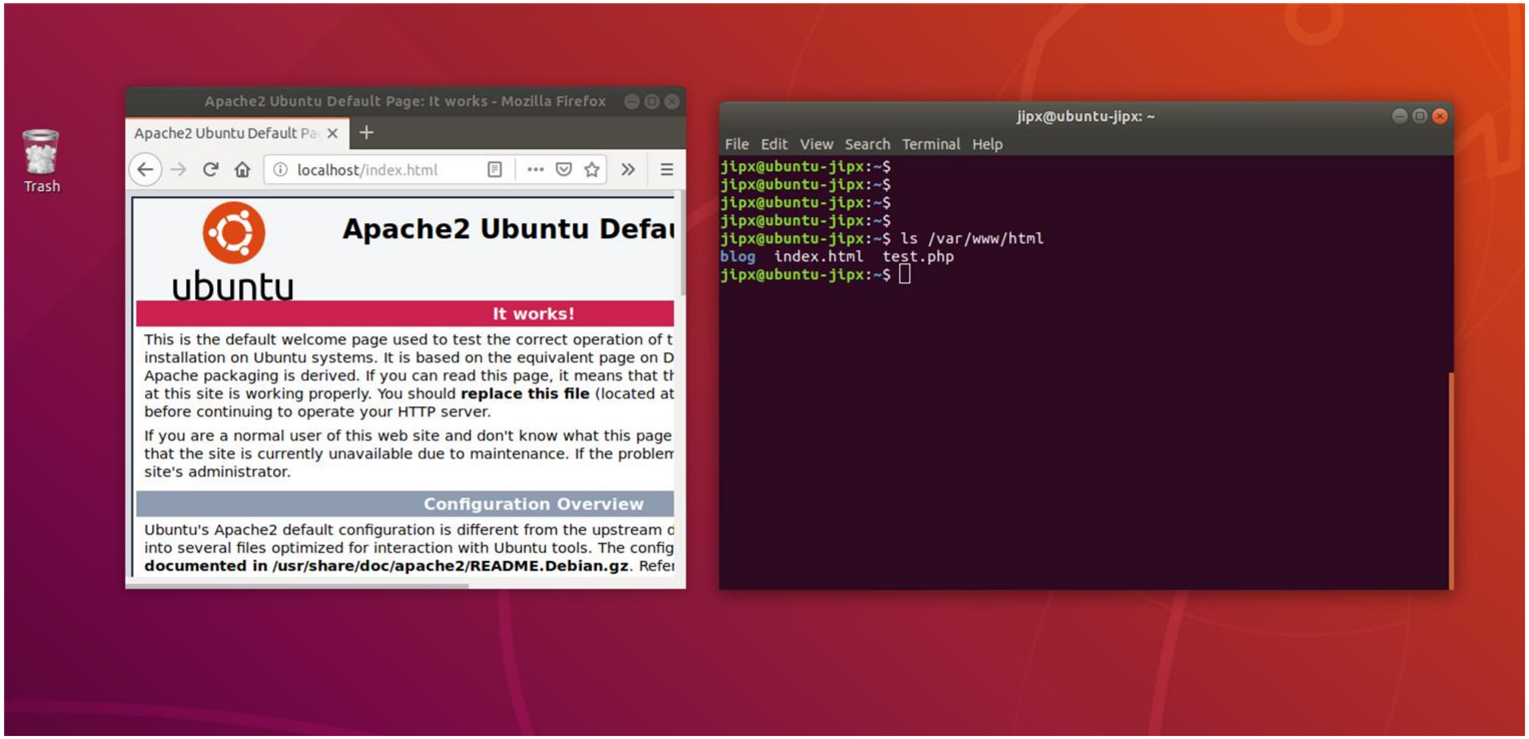
  Read more https://help.ubuntu.com/lts/serverguide/httpd.html.en-GB#http-directory-permissions

# DOCUMENT_ROOT /var/www/html



## Apache Environment

| Variable | Value |
|---|---|
| HTTP_HOST | localhost |
| HTTP_USER_AGENT | Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:67.0) Gecko/20100101 Firefox/67.0 |
| HTTP_ACCEPT | text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 |
| HTTP_ACCEPT_LANGUAGE | en-US,en;q=0.5 |
| HTTP_ACCEPT_ENCODING | gzip, deflate |
| HTTP_CONNECTION | keep-alive |
| HTTP_UPGRADE_INSECURE_REQUESTS | 1 |
| PATH | /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin |
| SERVER_SIGNATURE | <address>Apache/2.4.29 (Ubuntu) Server at localhost Port 80</address> |
| SERVER_SOFTWARE | Apache/2.4.29 (Ubuntu) |
| SERVER_NAME | localhost |
| SERVER_ADDR | 127.0.0.1 |
| SERVER_PORT | 80 |
| REMOTE_ADDR | 127.0.0.1 |
| DOCUMENT_ROOT | /var/www/html |
| REQUEST_SCHEME | http |
| CONTEXT_PREFIX | *no value* |
| CONTEXT_DOCUMENT_ROOT | /var/www/html |
| SERVER_ADMIN | webmaster@localhost |
| SCRIPT_FILENAME | /var/www/html/test.php |
| REMOTE_PORT | 51006 |
| GATEWAY_INTERFACE | CGI/1.1 |
| SERVER_PROTOCOL | HTTP/1.1 |
| REQUEST_METHOD | GET |
| QUERY_STRING | *no value* |
| REQUEST_URI | /test.php |
| SCRIPT_NAME | /test.php |


Trash
Apache2 Ubuntu Default Page: It works - Mozilla Firefox
Apache2 Ubuntu Default Pa
localhost/index.html
ubuntu
jipx@ubuntu-jipx: ~
File Edit View Search Terminal Help

Apache2 Ubuntu Defau

It works!

This is the default welcome page used to test the correct operation of t installation on Ubuntu systems. It is based on the equivalent page on D Apache packaging is derived. If you can read this page, it means that th at this site is working properly. You should **replace this file** (located at before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page that the site is currently unavailable due to maintenance. If the problem site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream d into several files optimized for interaction with Ubuntu tools. The config **documented in /usr/share/doc/apache2/README.Debian.gz**. Refe

```
jipx@ubuntu-jipx:~$
jipx@ubuntu-jipx:~$
jipx@ubuntu-jipx:~$
jipx@ubuntu-jipx:~$
jipx@ubuntu-jipx:~$ ls /var/www/html
blog  index.html  test.php
jipx@ubuntu-jipx:~$
```

**DocumentRoot /var/www/html**

then an access to http:/localhost/index.html refers to /var/www/html/index.htm

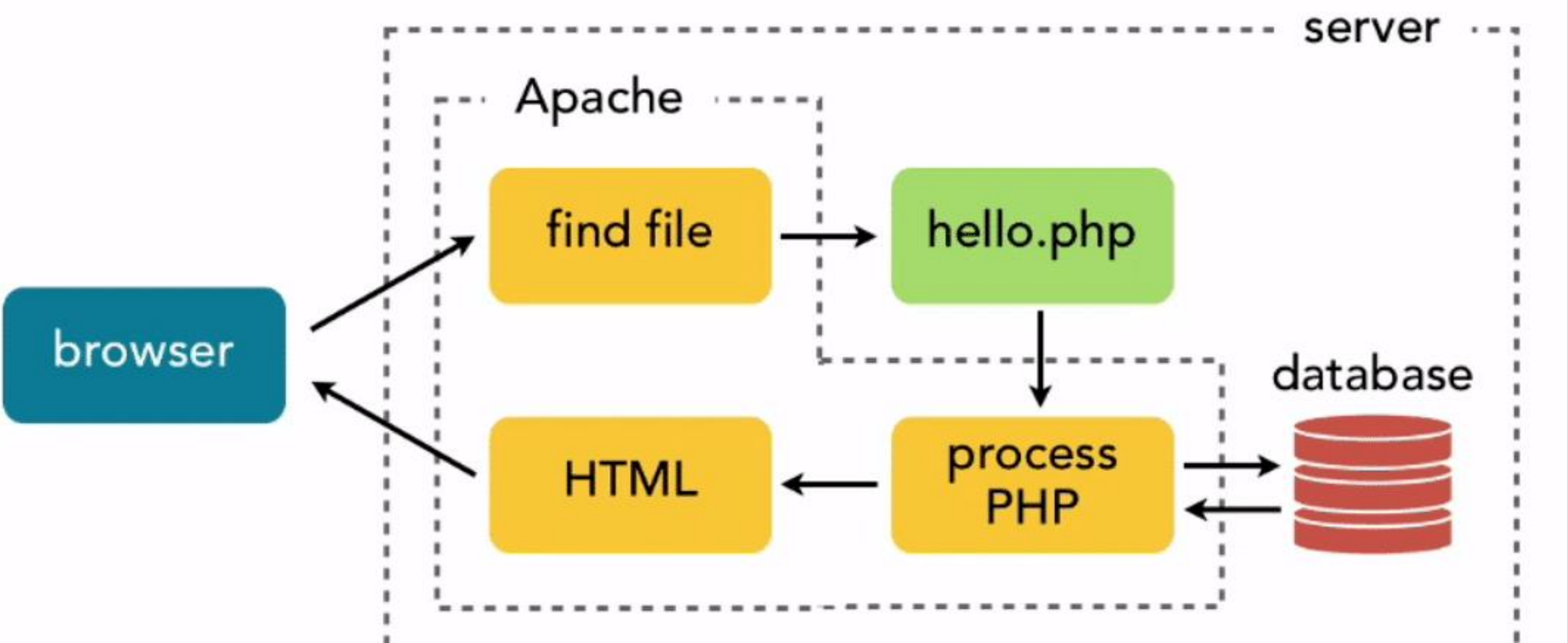
# What Apache needs to Run Php File?

Running Php files on Apache needs **mod_php** enabled on your server. It allows Apache to interpret .Php files. It has Php handlers that interpret the Php code in apache and send HTML to your web server.

If mod_php is enabled on your server, you will have a file named php.conf in /etc/httpd/conf.d/ directory. You can also check it with:

```
httpd -M | grep "php5_module"
```

https://geekflare.com/php-fpm-optimization/



# Install mysql

On Ubuntu 18.04, only the latest version of MySQL is included in the APT package repository by default. At the time of writing, that's MySQL 5.7

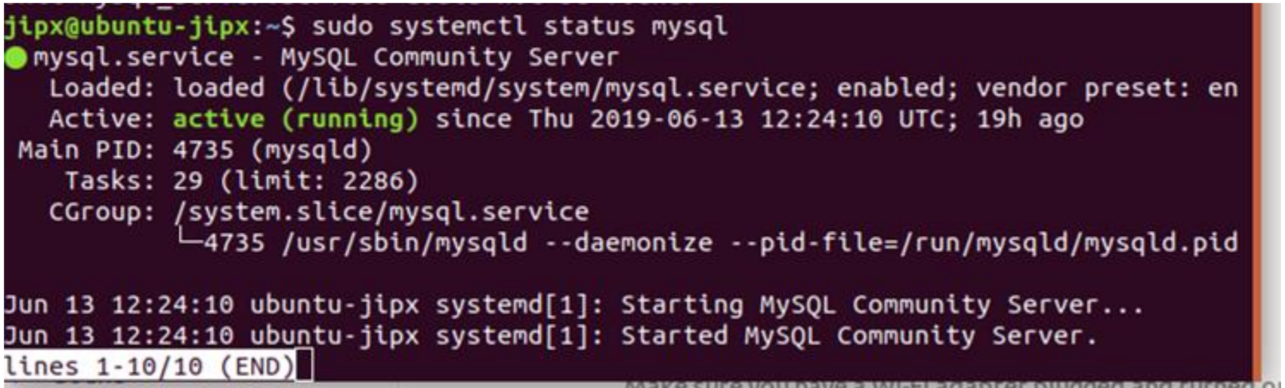
To install it, update the package index on your server with apt:

sudo apt update

Then install the default package:

sudo apt install mysql-server

## To check if the sql-service is running:
## sudo systemctl status mysql



```
jipx@ubuntu-jipx:~$ sudo systemctl status mysql
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: en
   Active: active (running) since Thu 2019-06-13 12:24:10 UTC; 19h ago
 Main PID: 4735 (mysqld)
    Tasks: 29 (limit: 2286)
   CGroup: /system.slice/mysql.service
           └─4735 /usr/sbin/mysqld --daemonize --pid-file=/run/mysqld/mysqld.pid

Jun 13 12:24:10 ubuntu-jipx systemd[1]: Starting MySQL Community Server...
Jun 13 12:24:10 ubuntu-jipx systemd[1]: Started MySQL Community Server.
lines 1-10/10 (END)
```

# Create a dedicated user for PHP application to conect to a specific database

1. Create a user and password for your MySQL database. Your WordPress installation uses these values to communicate with your MySQL database. Enter the following command, substituting a unique user name and password.

```
CREATE USER 'wordpress-user'@'localhost' IDENTIFIED BY 'your_strong_password';
```

```
2.     CREATE DATABASE `wordpress-db`;
```

3. Grant full privileges for your database to the WordPress user that you created earlier.

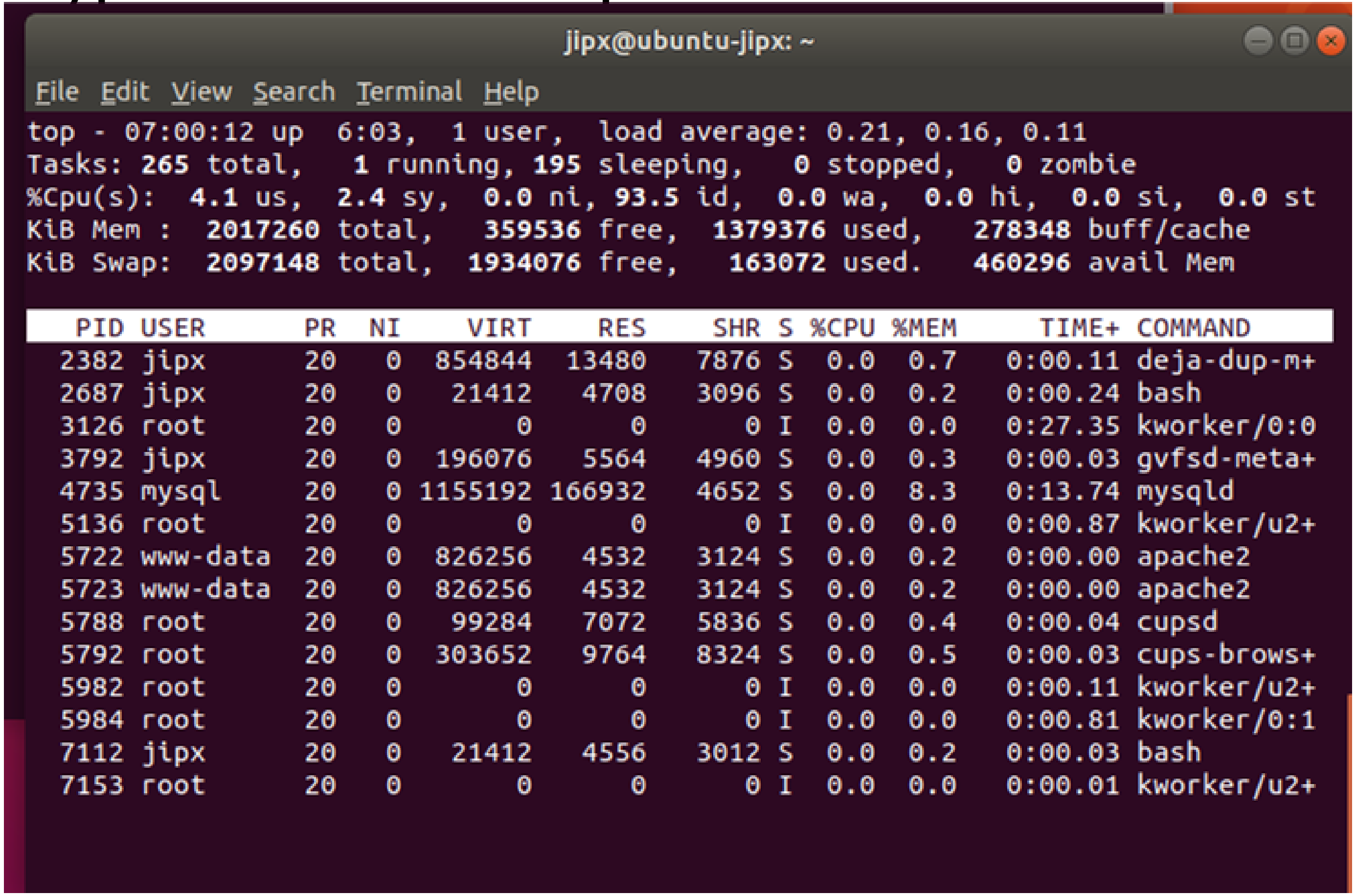
```
GRANT ALL PRIVILEGES ON `wordpress-db`.* TO "wordpress-user"@"localhost";
```

4. Flush the database privileges to pick up all of your changes.

5. `FLUSH PRIVILEGES;`

# What service/process is running

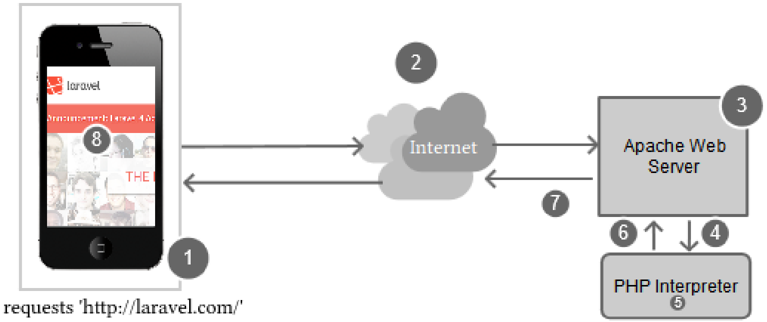
Type command top



```
top - 07:00:12 up  6:03,  1 user,  load average: 0.21, 0.16, 0.11
Tasks: 265 total,   1 running, 195 sleeping,   0 stopped,   0 zombie
%Cpu(s):  4.1 us,  2.4 sy,  0.0 ni, 93.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  2017260 total,   359536 free,  1379376 used,   278348 buff/cache
KiB Swap:  2097148 total,  1934076 free,   163072 used.   460296 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 2382 jipx      20   0  854844  13480   7876 S  0.0  0.7   0:00.11 deja-dup-m+
 2687 jipx      20   0   21412   4708   3096 S  0.0  0.2   0:00.24 bash
 3126 root      20   0       0      0      0 I  0.0  0.0   0:27.35 kworker/0:0
 3792 jipx      20   0  196076   5564   4960 S  0.0  0.3   0:00.03 gvfsd-meta+
 4735 mysql     20   0 1155192 166932   4652 S  0.0  8.3   0:13.74 mysqld
 5136 root      20   0       0      0      0 I  0.0  0.0   0:00.87 kworker/u2+
 5722 www-data  20   0  826256   4532   3124 S  0.0  0.2   0:00.00 apache2
 5723 www-data  20   0  826256   4532   3124 S  0.0  0.2   0:00.00 apache2
 5788 root      20   0   99284   7072   5836 S  0.0  0.4   0:00.04 cupsd
 5792 root      20   0  303652   9764   8324 S  0.0  0.5   0:00.03 cups-brows+
 5982 root      20   0       0      0      0 I  0.0  0.0   0:00.11 kworker/u2+
 5984 root      20   0       0      0      0 I  0.0  0.0   0:00.81 kworker/0:1
 7112 jipx      20   0   21412   4556   3012 S  0.0  0.2   0:00.03 bash
 7153 root      20   0       0      0      0 I  0.0  0.0   0:00.01 kworker/u2+
```

# Example



User requests 'http://laravel.com/'

**Step 1**

The user enters `http://laravel.com` into their browser and taps/hits 'enter'.

**Step 2**

After the user has tapped/hit 'enter', the browser sends the page request over the Internet to the web server.

**Step 3**

The web server *gets* the request and analyzes the request information. Apache realizes that we didn't specify a file, so it looks for a directory index and finds `index.php`.

**Step 4**

Since Apache knows to send files that end with the `.php` file extension to the PHP interpreter, it asks PHP to execute the file.

**Step 5**

In this step, PHP is executing the code contained in the `index.php` file from the request. During this step, PHP may interact with databases, the file system or make external API calls, amongst other things.

**Step 6**

After PHP has finished executing the `index.php` file, it sends the output back to Apache.

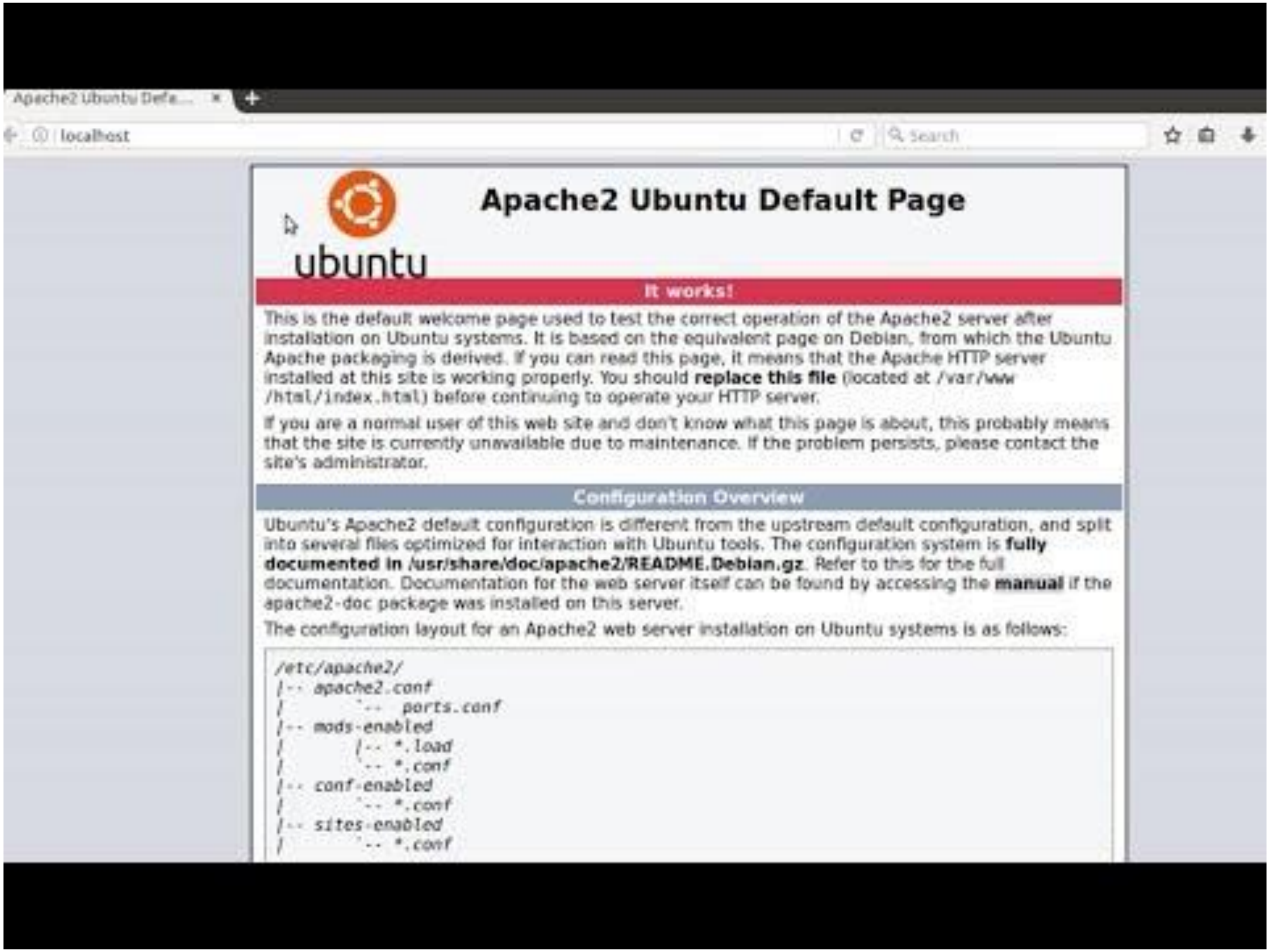
**Step 7**

Apache receives the output from PHP and sends it back over the Internet to a user's web browser. This is called the `web response`.

**Step 8**

The user's web browser receives the response from the server, and renders the web page on a computer or device.

# How to install apache, php, mysql & phpmyadmin



## Apache2 Ubuntu Default Page

**It works!**

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at /var/www/html/index.html) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

**Configuration Overview**

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

# Manaaging the Apache Process

Now that you have your web server up and running, let's go over some basic management commands.

To stop your web server, type:

```
sudo systemctl stop apache2
```

To start the web server when it is stopped, type:

```
sudo systemctl start apache2
```

To stop and then start the service again, type:

```
sudo systemctl restart apache2
```

If you are simply making configuration changes, Apache can often reload without dropping connections. To do this, use this command:

```
sudo systemctl reload apache2
```

By default, Apache is configured to start automatically when the server boots. If this is not what you want, disable this behavior by typing:

```
sudo systemctl disable apache2
```

To re-enable the service to start up at boot, type:

```
sudo systemctl enable apache2
```

# Write permission set for web master

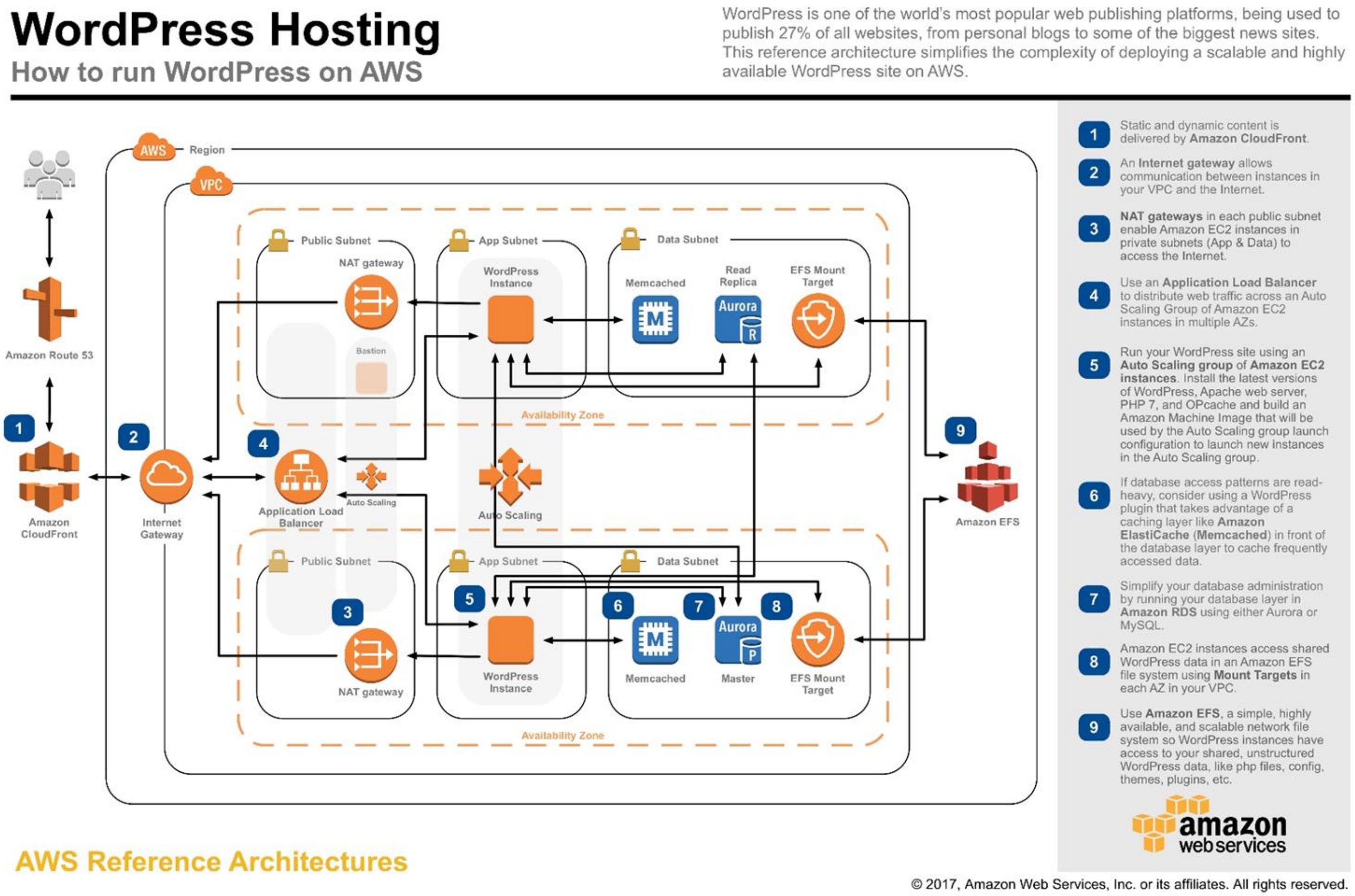
The following example grants shared write permission to `/var/www/html` to the group "webmasters".

```
sudo chgrp -R webmasters /var/www/html
sudo find /var/www/html -type d -exec chmod g=rwxs "{}" \;
sudo find /var/www/html -type f -exec chmod g=rw  "{}" \;
```

These commands recursively set the group permission on all files and directories in /var/www/html to read write and set user id. This has the effect of having the files and directories inherit their group and permission from their parrent. Many admins find this useful for allowing multiple users to edit files in a directory tree.

https://help.ubuntu.com/lts/serverguide/httpd.html.en-GB#http-directory-permissions

# Cloud Application



## WordPress Hosting

### How to run WordPress on AWS

WordPress is one of the world's most popular web publishing platforms, being used to publish 27% of all websites, from personal blogs to some of the biggest news sites. This reference architecture simplifies the complexity of deploying a scalable and highly available WordPress site on AWS.

1 Static and dynamic content is delivered by **Amazon CloudFront**.

2 An **Internet gateway** allows communication between instances in your VPC and the Internet.

3 **NAT gateways** in each public subnet enable Amazon EC2 instances in private subnets (App & Data) to access the Internet.

4 Use an **Application Load Balancer** to distribute web traffic across an Auto Scaling Group of Amazon EC2 instances in multiple AZs.

5 Run your WordPress site using an **Auto Scaling group** of **Amazon EC2 instances**. Install the latest versions of WordPress, Apache web server, PHP 7, and OPcache and build an Amazon Machine Image that will be used by the Auto Scaling group launch configuration to launch new instances in the Auto Scaling group.

6 If database access patterns are read-heavy, consider using a WordPress plugin that takes advantage of a caching layer like **Amazon ElastiCache** (**Memcached**) in front of the database layer to cache frequently accessed data.

7 Simplify your database administration by running your database layer in **Amazon RDS** using either Aurora or MySQL.

8 Amazon EC2 instances access shared WordPress data in an Amazon EFS file system using **Mount Targets** in each AZ in your VPC.

9 Use **Amazon EFS**, a simple, highly available, and scalable network file system so WordPress instances have access to your shared, unstructured WordPress data, like php files, config, themes, plugins, etc.

amazon web services

AWS Reference Architectures

# Summary

- HTTPd