WEB TECHNOLOGIES 1

Apache

Introduction

- A web server's job is basically to accept requests from clients and send responses to those requests.
- A web server gets a URL, translates it to a filename (for static requests), and sends that file back over the internet from the local disk.
- Or it translates it to a program name (for dynamic requests), executes it, and then sends the output of that program back over the internet to the requesting party.

APACHE

- Apache Web Server is designed to create web servers that have the ability to host one or more HTTP-based websites.
- The features include the ability to support multiple programming languages, server-side scripting, an authentication mechanism and database support.
- It is also widely used by web hosting companies for the purpose of providing shared/virtual hosting, as by default.
- Apache Web Server supports and distinguishes between different hosts that reside on the same machine.

XAMPP

- XAMPP is a free and open source cross-platform web server developed by Apache .
- XAMPP stands for :
    - X: Stands for Cross platform (Means you can install xampp on any OS)
    - A: Apache server (To run PHP script on local machine you need a server)
    - M: Mysql (To store data in database and to perform database operation you need Mysql)
    - P: PHP (Well known Scripting language to create dynamic website)
    - P: Perl programming language.

APACHE

- Although we call Apache a web server, it is not a physical server, but rather a software that runs on a server.
- Its job is to establish a connection between a server and the browsers of website visitors (Firefox, Google Chrome, Safari, etc.) while delivering files back and forth between them (client-server structure).
- Apache is a cross-platform software, therefore it works on both Unix and Windows servers.
- Apache has modules for security, caching, URL rewriting, password authentication, and more.

APACHE

- When a visitor wants to load a page on your website, for instance, the homepage or your “About Us” page, their browser sends a request to your server and Apache returns a response with all the requested files (text, images, etc.).
- The server and the client communicate through the HTTP protocol and Apache is responsible for the smooth and secure communication between the two machines.
- Apache is highly customizable, as it has a module-based structure.
- Modules allow server administrators to turn additional functionalities on and off.
- You can also set up your own server configurations through a file called .htaccess, which is an Apache configuration file supported with all Hostinger plans.

WHY APACHE ?

- Apache is the most popular web server available. The reasons behind its popularity are:
  - It is free to download and install.
  - It is open source: the source code is visible to anyone and everyone, which basically enables anyone to adjust the code, optimize it, and fix errors and security holes.
  - People can add new features and write new modules.
  - It suits all needs: Apache can be used for small websites of one or two pages, or huge websites of hundreds and thousands of pages.
  - Thus serving millions of regular visitors each month. It can serve both static and dynamic content.

Apache server

- The Apache server offers a number of services that clients might make use of.
- These services are offered using various protocols through different ports, and include:
  - Hypertext transfer protocol (HTTP), typically through port 80.
  - Simple mail transfer protocol (SMTP), typically through port 25.
  - Domain name service (DNS) for mapping domain names to their corresponding IP addresses, generally through port 53.
  - File transfer protocol (FTP) for uploading and downloading files, usually through port 21.

Apache Pros and Cons

- An Apache web server can be an excellent choice to run your website on a stable and versatile platform. However, it also comes with some disadvantages you need to pay attention to.
- **Pros**:
  - Open-source and free, even for commercial use.
  - Reliable, stable software.
  - Frequently updated, regular security patches.
  - Flexible due to its module-based structure.
  - Easy to configure, beginner-friendly.
  - Cross-platform (works on both Unix and Windows servers).
  - Works out of the box with WordPress sites.
  - Huge community and easily available support in case of any problem.
- **Cons**:
  - Performance problems on extremely traffic-heavy websites.
  - Too many configuration options can lead to security vulnerabilities.

How Apache Works

- Apache's main role is all about communication over networks, and it uses the TCP/IP protocol.
- The Apache server is set up to run through configuration files, in which directives are added to control its behavior.
- In its idle state, Apache listens to the IP addresses identified in its config file (HTTPd.conf).
- Whenever it receives a request, it analyzes the headers, applies the rules specified for it in the Config file, and takes action.

General Structure

- Apache can be installed on a variety of operating systems.
- The Apache binary running under UNIX is called *HTTPd* (short for HTTP daemon), and under win32 is called *Apache.exe*.
- A hosted website will typically have four main directories:
  - *htdocs*
  - *Conf*
  - *logs*
  - *cgi-bin*

General Structure

- **Htdocs**
  - The default Apache web server document directory,it is the public directory whose contents are usually available for clients connecting through the web.
  - It contains all static pages and dynamic content to be served once an HTTP request for them is received.
  - Since files and sub-directories under htdocs are available to the public, correct handling of file permissions is of great importance so as not to compromise the server's safety and security.

General Structure

- **cgi-bin**
  - Its the directory where CGI scripts are kept.
  - The CGI (Common Gateway Interface) defines a way for a web server to interact with external content-generating programs, which are often referred to as CGI programs or CGI scripts.
  - These are programs or shell scripts that are written to be executed by Apache on behalf of its clients.
- **conf**
  - The directory where all server configuration files are located.
  - Configuration files are plain text files where directives are added to control the web server's behavior and functionality.
  - Placed on a separate line, and the hash (#) key

General Structure

- **logs**
  - The directory where server logs are kept, and includes Apache access logs and error logs.
  - The Apache HTTP Server provides a variety of different mechanisms for logging everything that happens on it.
  - From the initial request, through the URL mapping process, to the final resolution of the connection, including any errors that may have occurred in the process.
  - Third-party modules may provide logging capabilities, or inject entries into the existing log files, and applications such as PHP scripts, or other handlers, may send messages to the server error log.

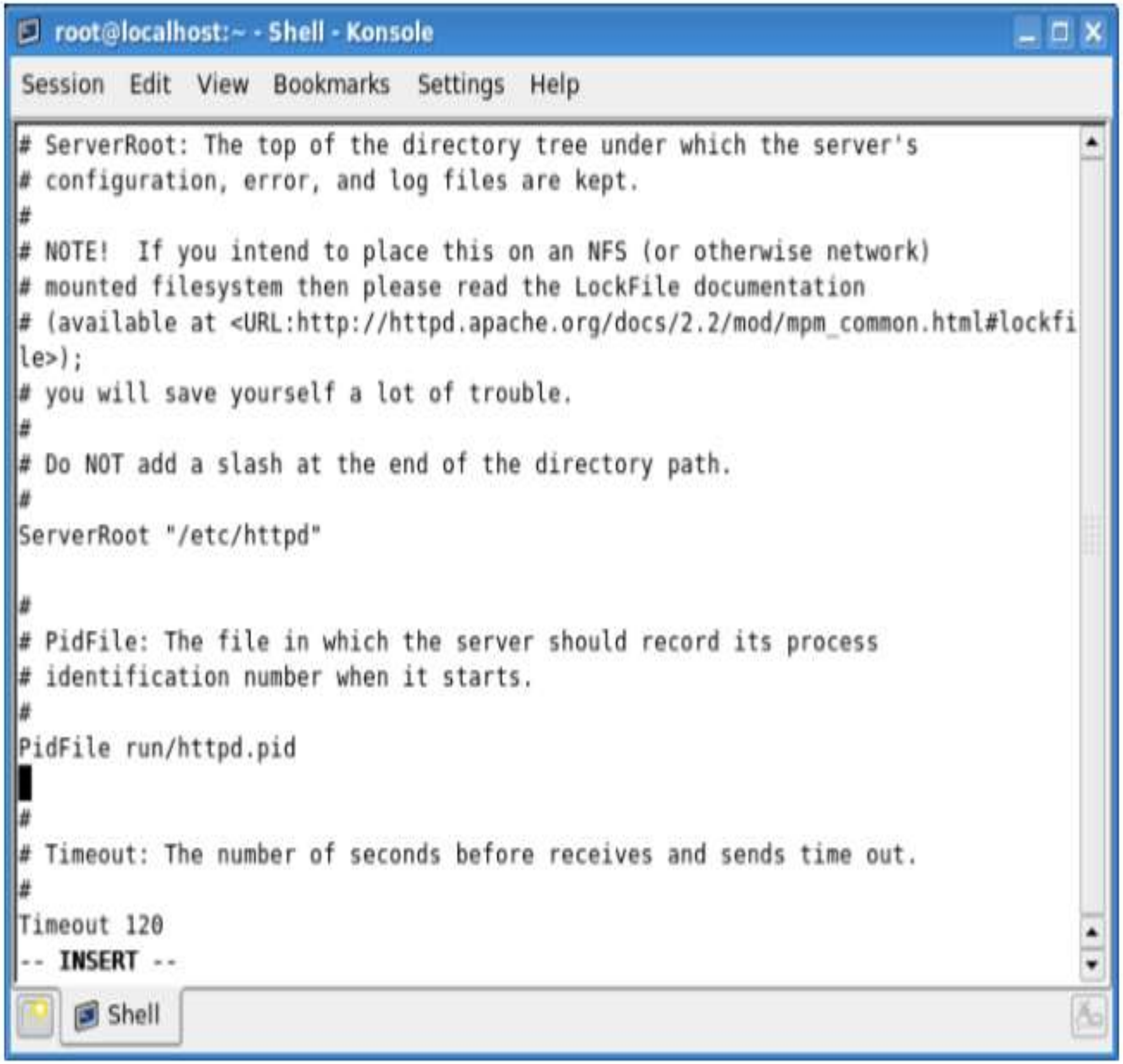
HTTPD- *HYPERTEXT TRANSFER PROTOCOL DAEMON*

- Apache **HTTPD** is an HTTP server daemon produced by the Apache Foundation.
- **HTTP Daemon** is a software program that runs in the background of a web server and waits for the incoming server requests. It listens to one or more TCP ports (usually port 80).
- The daemon answers the request automatically and serves the hypertext and multimedia documents over the Internet using HTTP.
- It is a piece of software that listens for network requests and responds to them. It is open source and many entities **use** it to host their websites.
- **HTTPD** can be extended with modules. A very common way to use httpd is with the mod_php extension which allows it to parse PHP code.

SERVERROOT

- ServerRoot is the path to the server's configuration, error and log files.
- It is possible to change this path, provided all the necessary files are copied to the new location accordingly.
- The **default location** is **/etc/httpd**.
- The file is rich with easy-to-understand comments.

SERVERROOT



root@localhost:~ - Shell - Konsole
Session Edit View Bookmarks Settings Help
Shell

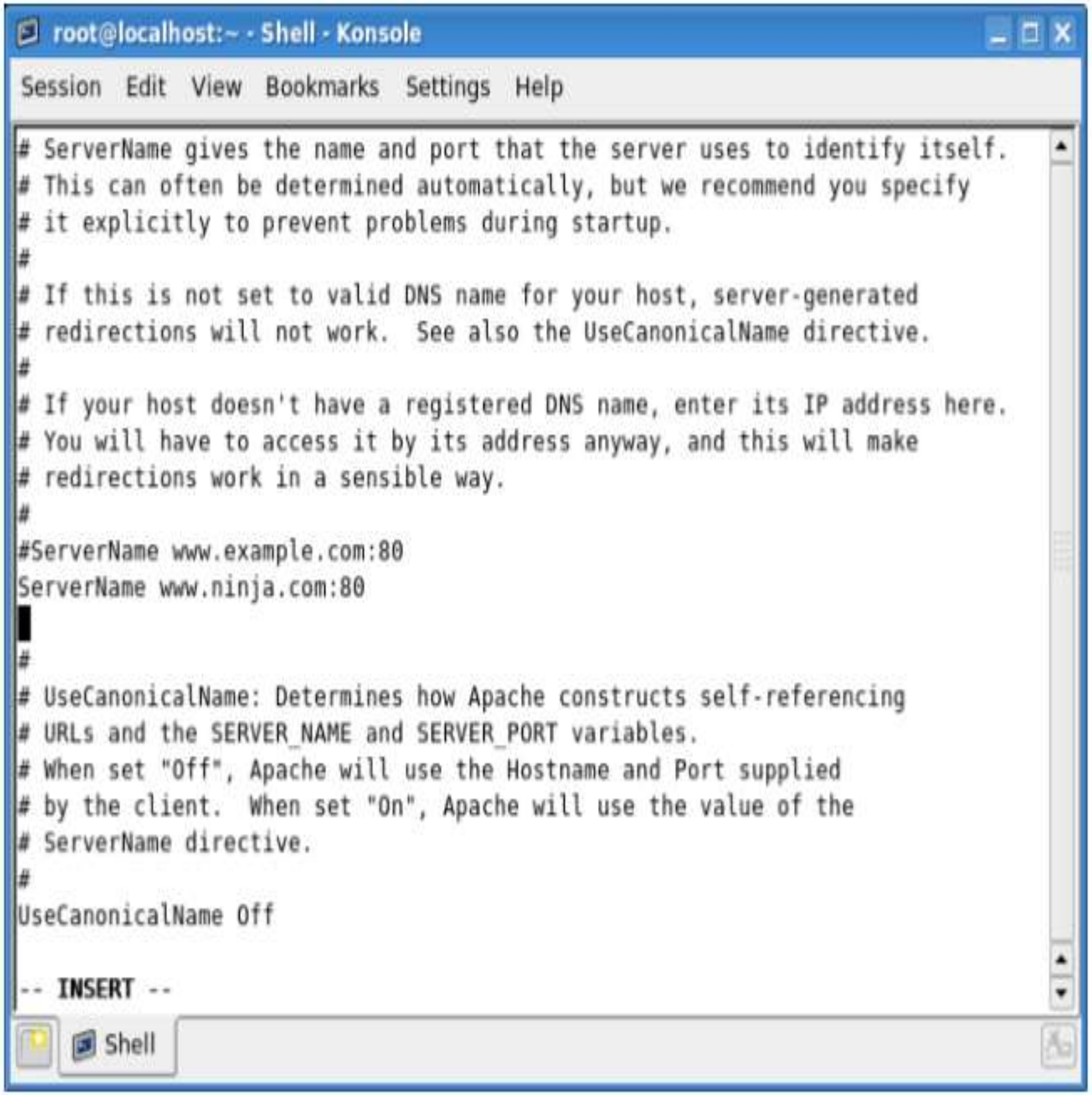
```
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
# NOTE!  If you intend to place this on an NFS (or otherwise network)
# mounted filesystem then please read the LockFile documentation
# (available at <URL:http://httpd.apache.org/docs/2.2/mod/mpm_common.html#lockfi
le>);
# you will save yourself a lot of trouble.
#
# Do NOT add a slash at the end of the directory path.
#
ServerRoot "/etc/httpd"

#
# PidFile: The file in which the server should record its process
# identification number when it starts.
#
PidFile run/httpd.pid

#
# Timeout: The number of seconds before receives and sends time out.
#
Timeout 120
-- INSERT --
```

PIDFILE

- PidFile is the process identification number for the httpd.
- This process number is important, because Apache spawns numerous child processes when running to accommodate the web traffic.
- It allows you to monitor and manipulate your server processes.

SERVERNAME

- ServerName gives the name and port that the server uses to identify itself.
- This option is to set to your DNS hostname, or IP address (to find out your IP address, run command "ipconfig"), or your computer name, or "localhost" (localhost is meant for local loop-back testing only) .


root@localhost:~ - Shell - Konsole
Session Edit View Bookmarks Settings Help
Shell

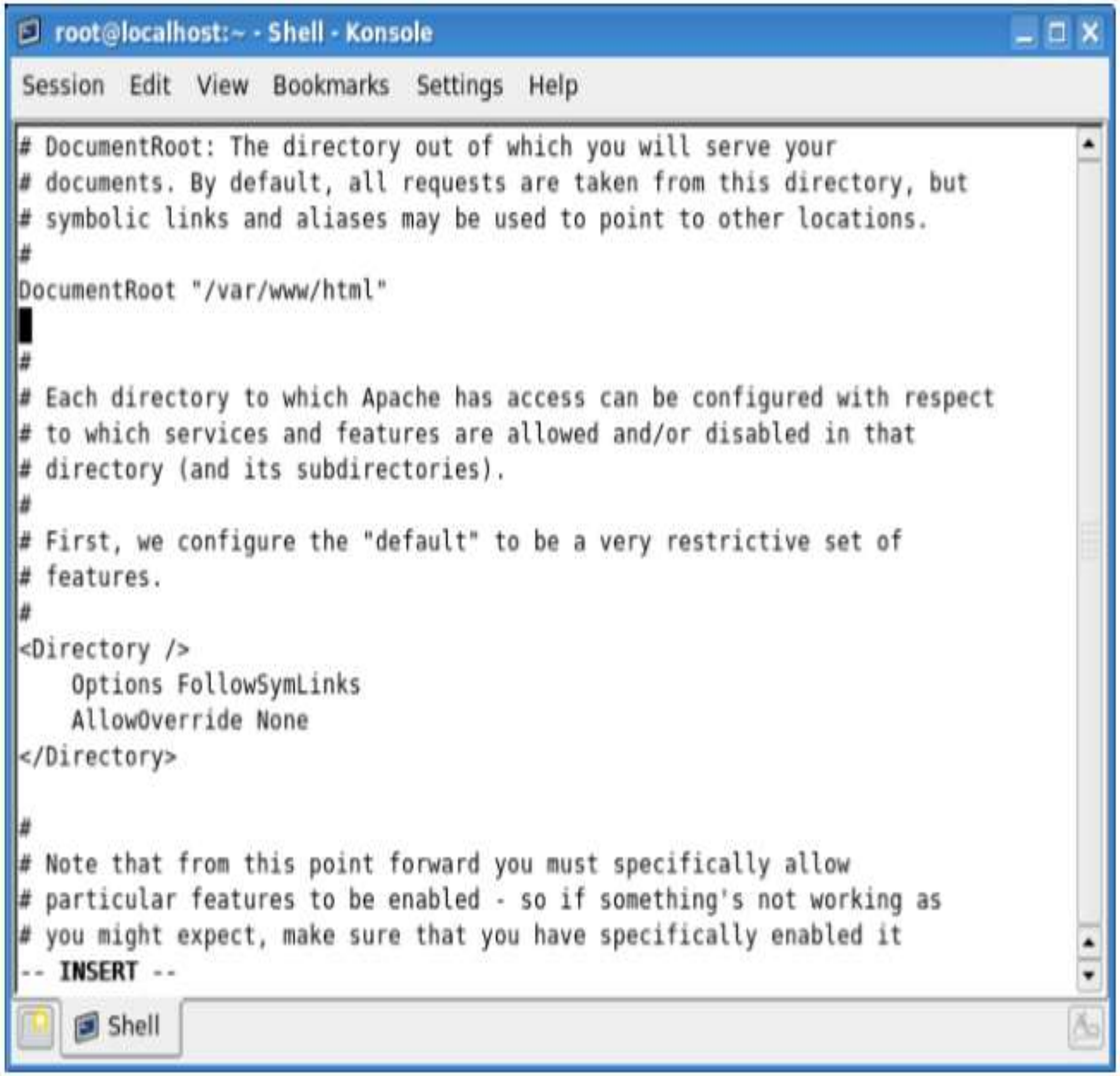
```
# ServerName gives the name and port that the server uses to identify itself.
# This can often be determined automatically, but we recommend you specify
# it explicitly to prevent problems during startup.
#
# If this is not set to valid DNS name for your host, server-generated
# redirections will not work.  See also the UseCanonicalName directive.
#
# If your host doesn't have a registered DNS name, enter its IP address here.
# You will have to access it by its address anyway, and this will make
# redirections work in a sensible way.
#
#ServerName www.example.com:80
ServerName www.ninja.com:80

#
# UseCanonicalName: Determines how Apache constructs self-referencing
# URLs and the SERVER_NAME and SERVER_PORT variables.
# When set "Off", Apache will use the Hostname and Port supplied
# by the client.  When set "On", Apache will use the value of the
# ServerName directive.
#
UseCanonicalName Off

-- INSERT --
```

/ETC/HOSTS FILE

- The hosts file allows easy matching of names to IP addresses.
- In general, using the hosts file is a good way of testing your IP-to-name (or vice versa) configurations before committing these changes into a production environment.

DocumentRoot

- DocumentRoot tells you where your web documents (html files, images etc) should be located.
- It is possible to reference files in other directories using aliases and symbolic links.



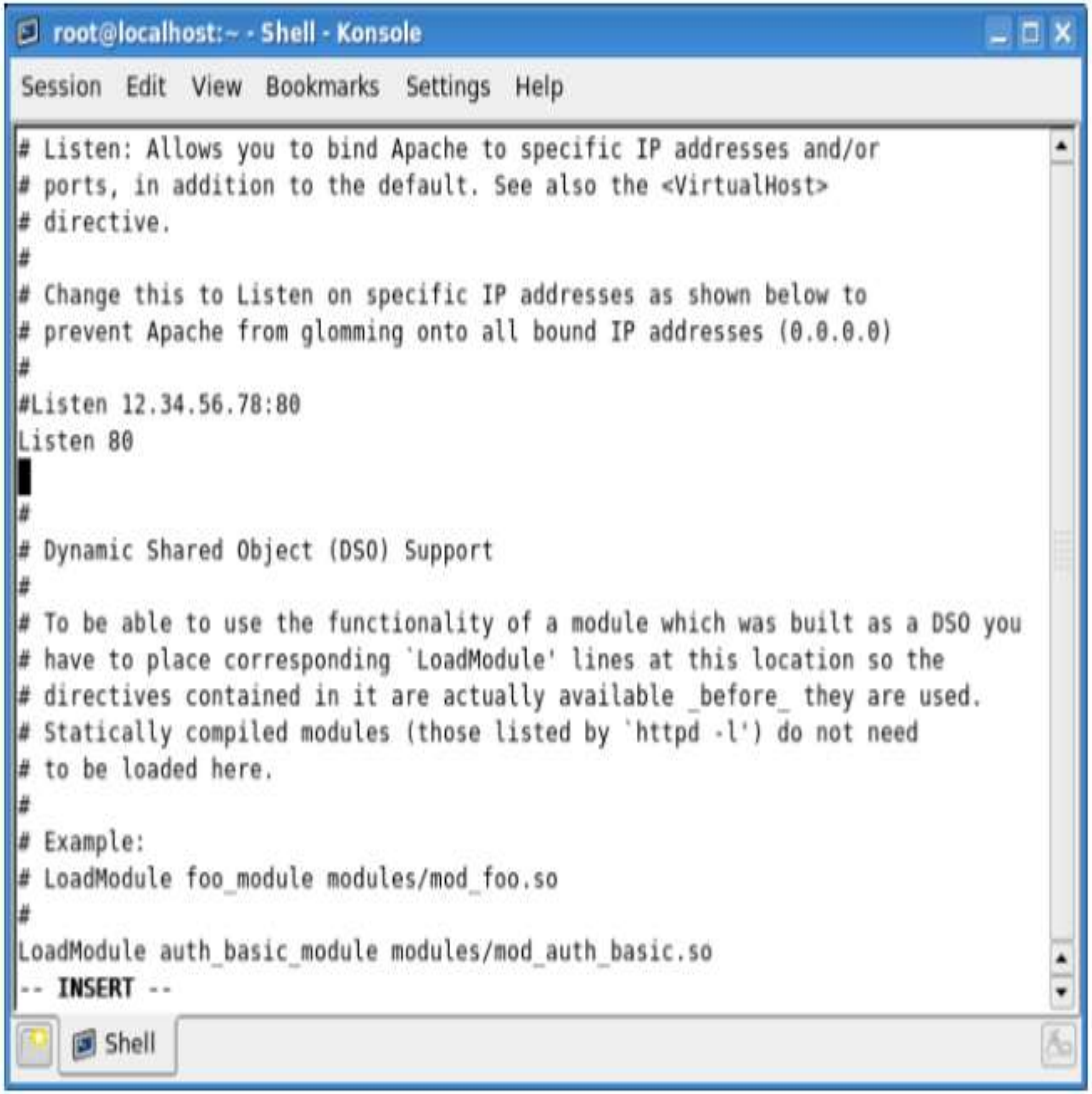
```
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "/var/www/html"

#
# Each directory to which Apache has access can be configured with respect
# to which services and features are allowed and/or disabled in that
# directory (and its subdirectories).
#
# First, we configure the "default" to be a very restrictive set of
# features.
#
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>

#
# Note that from this point forward you must specifically allow
# particular features to be enabled - so if something's not working as
# you might expect, make sure that you have specifically enabled it
-- INSERT --
```

ErrorLog

- ErrorLog tells you where the log containing all server errors is located.
- This file is critical for debugging and solving server misconfiguration problems and for proper traffic shaping.
- By default, all messages with the value of warning(warn) and higher will be logged. This is described in the Log Level directive just below.
- The **default location** is **logs/error_log**.
- Please note that this is relative to the ServerRoot. Therefore, our log file is /etc/httpd/logs/error_log.

LISTEN

- The Listen command tells the Webserver what ports to use for incoming connections.
- By default, port 80 is used, although any one or several can be used.
- The accepted conventions calls for using port 80 for non-secure web communications (without any encryption of traffic).
- Secure web communications are normally handled on port 443.


root@localhost:~ - Shell - Konsole
Session Edit View Bookmarks Settings Help
Shell

```
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, in addition to the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses (0.0.0.0)
#
#Listen 12.34.56.78:80
Listen 80

#
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which was built as a DSO you
# have to place corresponding `LoadModule' lines at this location so the
# directives contained in it are actually available _before_ they are used.
# Statically compiled modules (those listed by `httpd -l') do not need
# to be loaded here.
#
# Example:
# LoadModule foo_module modules/mod_foo.so
#
LoadModule auth_basic_module modules/mod_auth_basic.so
-- INSERT --
```